

เก่ง Django ได้ใน 1 วัน

เอกสิทธิ์ ศรีสุขะ

Appcode Training

เนื้อหา

- การติดตั้ง
- การสร้าง / รันโปรเจกต์
- วิวและเทมเพลต
- โมเดล
- จัดการข้อมูลด้วย Admin
- การแสดงรายการ & แบ่งหน้า
- ฟอรัม
- คลาสเบสวิว
- ล็อกอิน/ล็อกเอาต์
- การส่งเมล
- การทำ Rest API
- การติดตั้งลงเซิร์ฟเวอร์

ทำไมเลือก Django ?

- ได้งานเร็ว
- เครื่องมือครบ
- ประสิทธิภาพดี
- เอกสารเยอะ
- รองรับคลาวด์

ใครใช้ Django บ้าง

Instagram, Pinterest, Disqus, Bitbucket, NASA, Eventbrite, Spotify
... และบริษัทในประเทศไทยอีกจำนวนมาก...

ชั่วโมงที่

1

การติดตั้ง

การติดตั้ง

- Python3
- Virtualenv
- PostgreSQL (ติดตั้งภายหลังได้)
- Visual Studio Code

การติดตั้ง Python

ดาวน์โหลดและติดตั้ง Python (3.x) ได้จาก <https://www.python.org>

การติดตั้ง Python

ตรวจสอบด้วยคำสั่ง

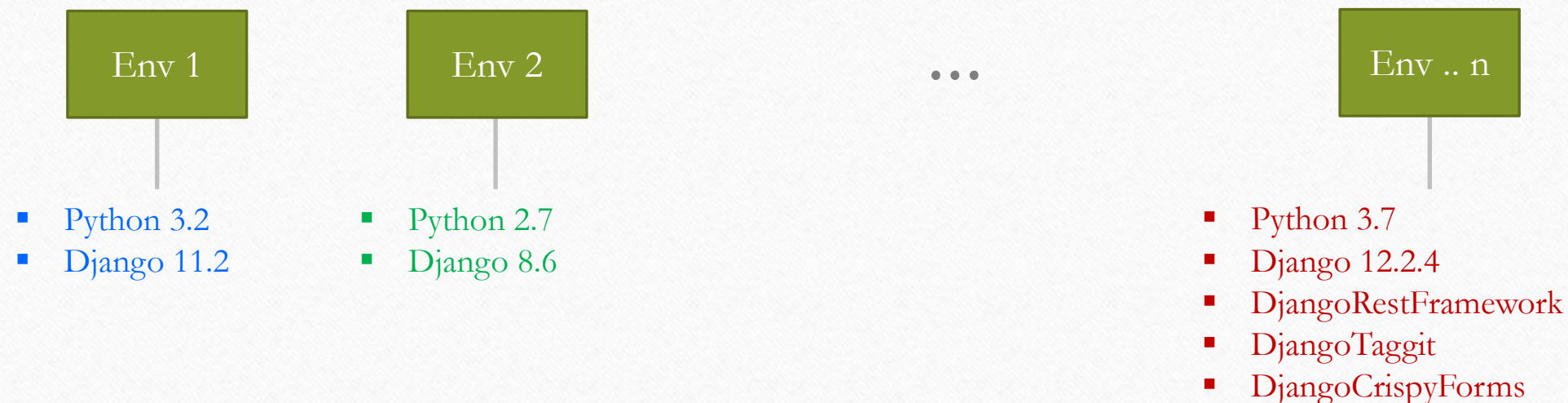
```
python -V
```

และ

```
pip -V
```


การติดตั้ง Virtualenv

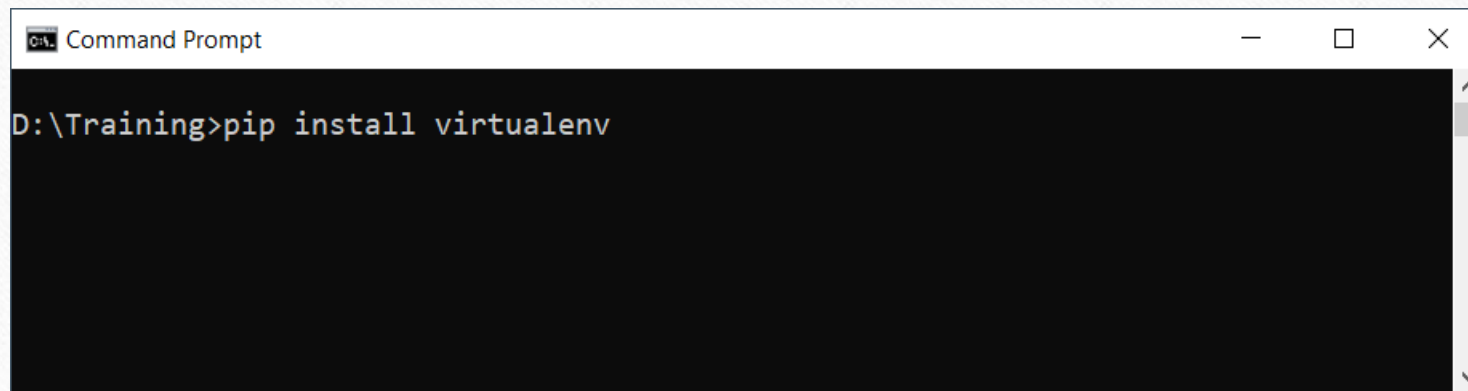
ช่วยจำลองสภาพแวดล้อมหลายๆ ตัวในเครื่องเดียว



การติดตั้ง Virtualenv (ต่อ)

ติดตั้งโดยใช้คำสั่ง `pip install <ชื่อไลบรารี>`

```
pip install virtualenv
```

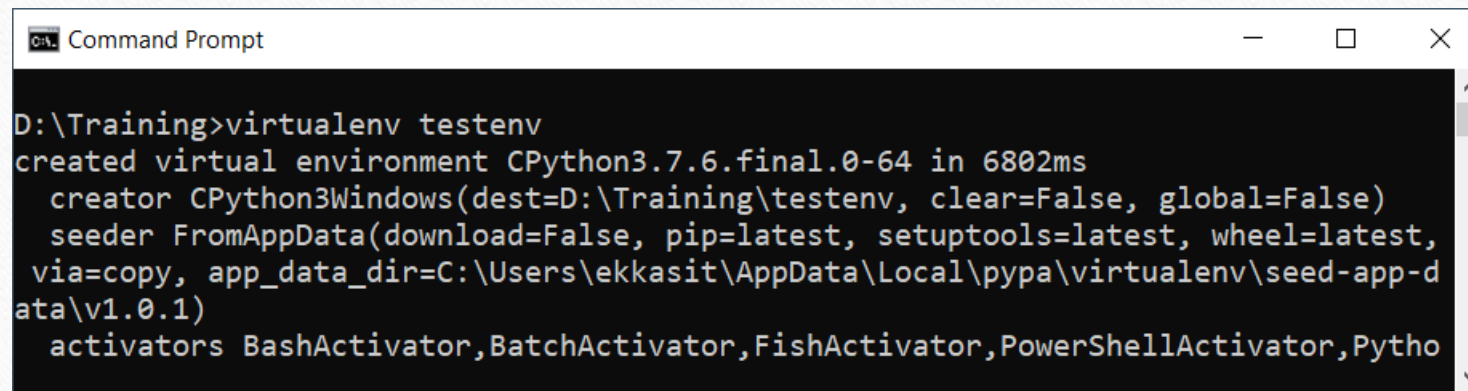


```
Command Prompt
D:\Training>pip install virtualenv
```

การติดตั้ง Virtualenv (ต่อ)

การสร้าง environment จะใช้คำสั่ง **virtualenv** <ชื่อที่ต้องการ>

```
virtualenv testenv
```



```
Command Prompt
D:\Training>virtualenv testenv
created virtual environment CPython3.7.6.final.0-64 in 6802ms
  creator CPython3Windows(dest=D:\Training\testenv, clear=False, global=False)
  seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest,
via=copy, app_data_dir=C:\Users\ekkasit\AppData\Local\pypa\virtualenv\seed-app-d
ata\v1.0.1)
  activators BashActivator,BatchActivator,FishActivator,PowerShellActivator,Pytho
```


การติดตั้ง Virtualenv (ต่อ)

- เปิดการใช้งาน ใช้คำสั่ง **activate**

```
$ Scripts\activate
```

- จบการทำงาน ใช้คำสั่ง **deactivate**

```
(testenv) $ deactivate
```

การติดตั้ง Django Framework

- เปิดใช้งาน virtual environment ให้เรียบร้อย
- ติดตั้งจะใช้คำสั่ง **pip install django**

```
(testenv) $ pip install django
```

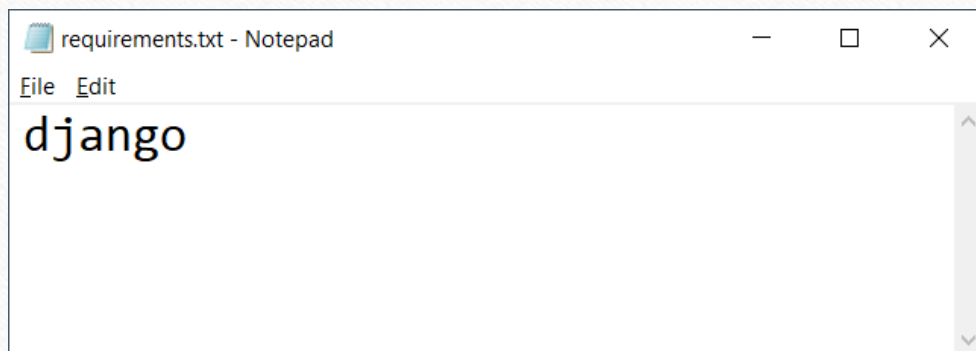
หรือ หากต้องการระบุเวอร์ชัน

```
(testenv) $ pip install django==3.0.8
```

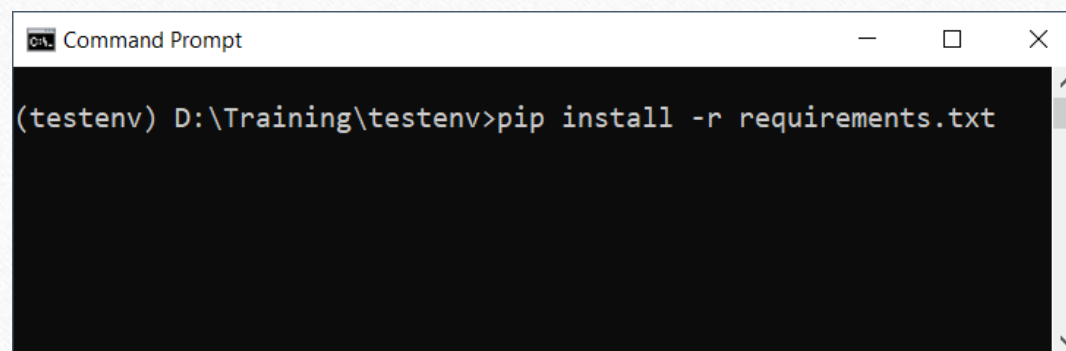
ใส่เท่ากับ
แล้วตามด้วยเวอร์ชัน

การติดตั้ง Django Framework (ต่อ)

เราแนะนำให้ติดตั้งผ่าน **requirements.txt**



```
requirements.txt - Notepad
File Edit
django
```



```
C:\> Command Prompt
(testenv) D:\Training\testenv>pip install -r requirements.txt
```

```
pip install -r requirements.txt
```


การสร้างโปรเจกต์

ใช้คำสั่ง `django-admin startproject` <ชื่อโปรเจกต์>

```
django-admin startproject testproject
```

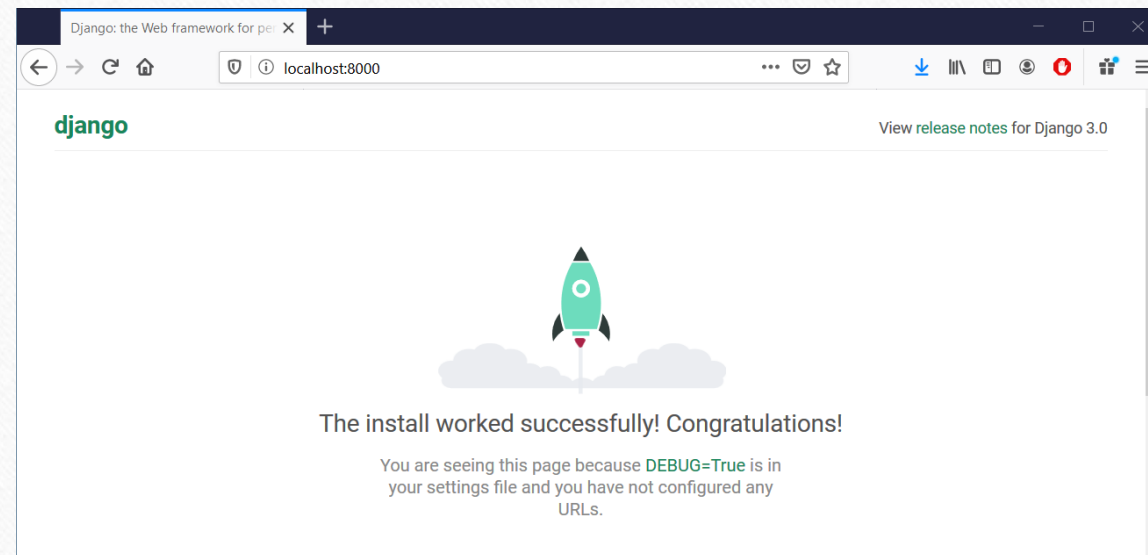
การรันโปรเจกต์

การเปิดเซิร์ฟเวอร์จะใช้คำสั่ง

```
python manage.py runserver
```

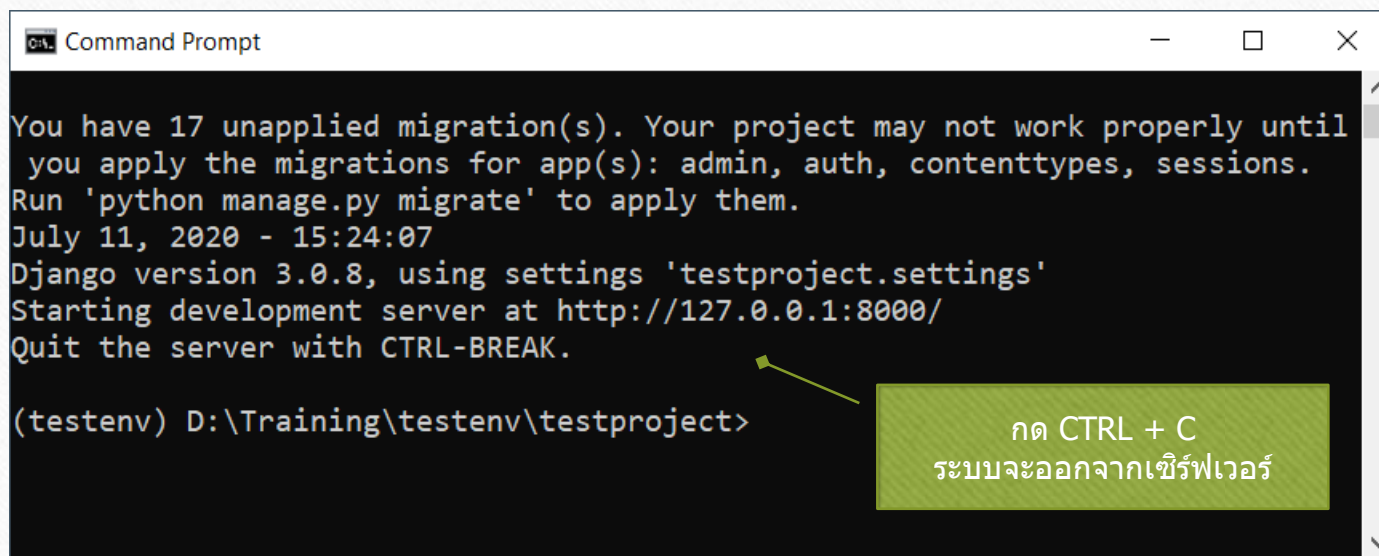
การรันโปรเจกต์ (ต่อ)

ทดสอบด้วย <http://localhost:8000>



การรันโปรเจกต์ (ต่อ)

หากต้องการปิดเซิร์ฟเวอร์ ให้กด **CTRL + C**



```
Command Prompt

You have 17 unapplied migration(s). Your project may not work properly until
you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 11, 2020 - 15:24:07
Django version 3.0.8, using settings 'testproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

(testenv) D:\Training\testenv\testproject>
```

กด CTRL + C
ระบบจะออกจากเซิร์ฟเวอร์

ติดตั้งเครื่องมือ

- ติดตั้ง VSCode
- ติดตั้งปลั๊กอินสำหรับ VSCode

ติดตั้งเครื่องมือ : VSCode

ดาวน์โหลดและติดตั้ง Visual Studio Code ได้ที่
<https://code.visualstudio.com>

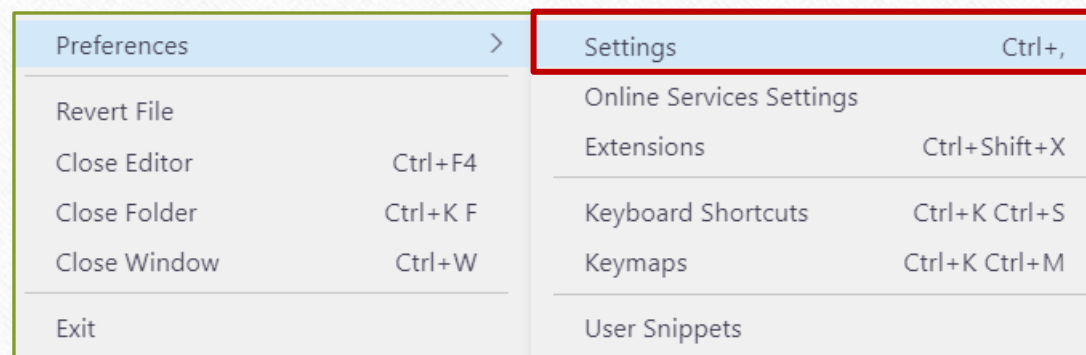
ติดตั้งเครื่องมือ : VSCode Plugins

ติดตั้งปลั๊กอินที่ใช้ ตามตาราง

ชื่อปลั๊กอิน	ผู้พัฒนา	หน้าที่
Python	Microsoft	รองรับการเขียนโปรแกรม Python และ Virtualenv
Path Intellisense	Christian Kohler	ช่วยในการค้นหา Path ที่เราต้องการ
EditorConfig for VS Code	EditorConfig	ช่วยจัดรูปแบบของโค้ด ตามที่เรากำหนด

ติดตั้งเครื่องมือ : VSCode Settings

ตั้งค่า VSCode เพื่อให้รู้จักกับ **Virtualenv** ดังนี้



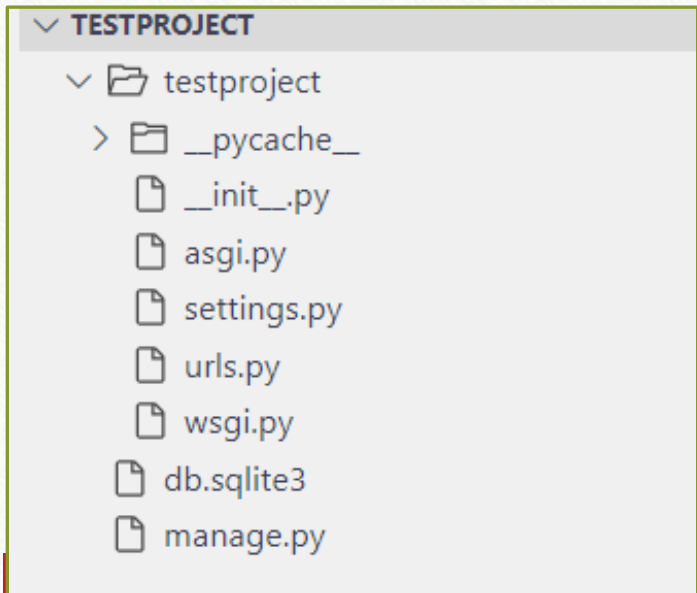
คลิกที่ Settings

settings.json

```
{  
  "python.pythonPath": "${workspaceFolder}../../Scripts/python.exe",  
  "terminal.integrated.shell.windows": "C:\\Windows\\System32\\cmd.exe"  
}
```

โครงสร้างไฟล์

โครงสร้างไฟล์ของโปรเจกต์



โฟลเดอร์/ไฟล์	หน้าที่
manage.py	ไฟล์สำหรับรับคำสั่งจากผู้ใช้
db.sqlite3	ไฟล์ฐานข้อมูล SQLite
testproject	โฟลเดอร์ส่วนกลางของโปรเจกต์
testproject/settings.py	ไฟล์ตั้งค่าโปรเจกต์
testproject/urls.py	ไฟล์กำหนด URL และ routing
testproject/wsgi.py	ไฟล์สำหรับรองรับ WSGI เซิร์ฟเวอร์
testproject/asgi.py	ไฟล์สำหรับรองรับ ASGI เซิร์ฟเวอร์ (ทำงานแบบ asynchronous)

รู้จักกับแอป

- ในหนึ่งโปรเจกต์ จะมีหลายๆ แอปอยู่ภายใน
- ยกตัวอย่างระบบ CMS
 - แอปสำหรับงานทั่วไป (app)
 - แอปสำหรับข่าว (news)
 - แอปสำหรับสนทนา (chats)
 - แอปสำหรับตอบปัญหา (faq)

การสร้างแอป

- เราสร้างแอปสำหรับงานทั่วไป ชื่อว่า “myapp”
- ใช้คำสั่ง `python manage.py startapp` <ชื่อแอป>

```
python manage.py startapp myapp
```

การสร้างแอป (ต่อ)

เมื่อสร้างแล้ว ให้ทำการ “ผูกแอป” ใน `settings.py` เพื่อให้ระบบรู้จัก

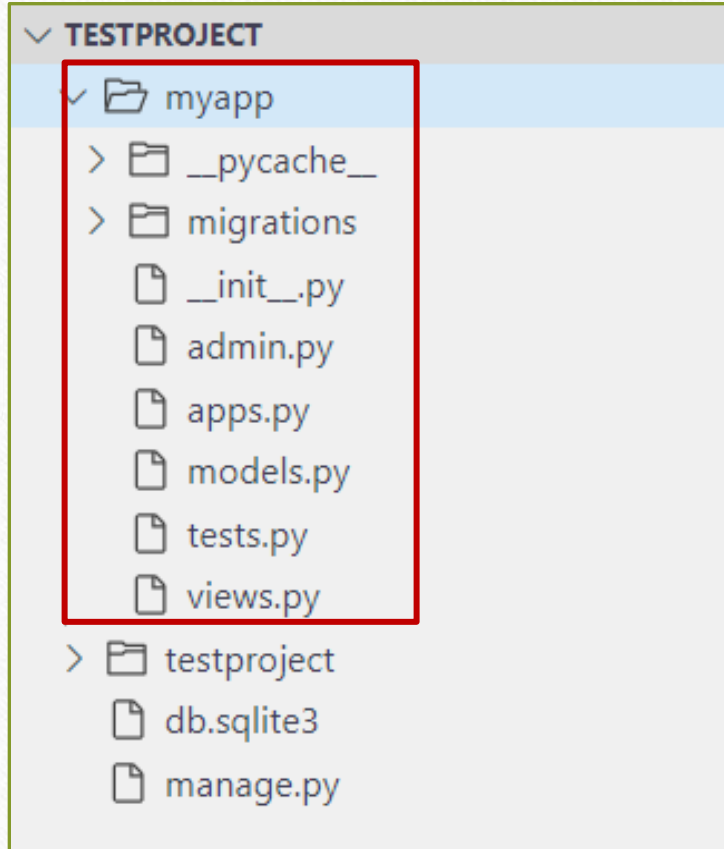
```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'myapp',  
]
```

testproject/settings.py

เพิ่มบรรทัดนี้ลงไป



โครงสร้างแอป



โฟลเดอร์/ไฟล์	หน้าที่
admin.py	ไฟล์สำหรับตั้งค่าหน้า admin
apps.py	ไฟล์สำหรับตั้งค่าของแอป (เฉพาะแอป)
models.py	ไฟล์โมเดลของฐานข้อมูล
tests.py	ไฟล์สำหรับเขียน unit test
views.py	ไฟล์สำหรับรับส่งข้อมูลในหน้าจอ
migrations	โฟลเดอร์ที่เก็บไฟล์ต่างๆ ของการ migration กับฐานข้อมูล



การตั้งค่าโปรเจกต์

พารามิเตอร์	หน้าที่
SECRET_KEY	ข้อมูลคีย์ที่ใช้เข้ารหัสข้อมูลของระบบ (อย่าให้รู้ว่ไหน)
DEBUG	DEBUG = True → ใช้ตอนพัฒนาโปรแกรม DEBUG = False → ใช้ตอนนำระบบขึ้น production
ALLOW_HOSTS	รายชื่อโฮสต์ทั้งหมดของระบบ
INSTALLED_APPS	รายชื่อแอปที่ลงทะเบียน
MIDDLEWARE	รายชื่อ middleware ของระบบ (ตัวกั้นระหว่าง request / response)
TEMPLATES	กำหนดโฟลเดอร์เทมเพลต เพื่อแสดงผล
DATABASES	กำหนดการเชื่อมต่อฐานข้อมูล
LANGUAGE, TIMEZONE, I18N	ตั้งค่าภาษา, เขตเวลา และการแปลภาษา
STATIC_URL	ตั้งค่าโฟลเดอร์ที่เก็บ static ไฟล์ เช่น JavaScript และ CSS

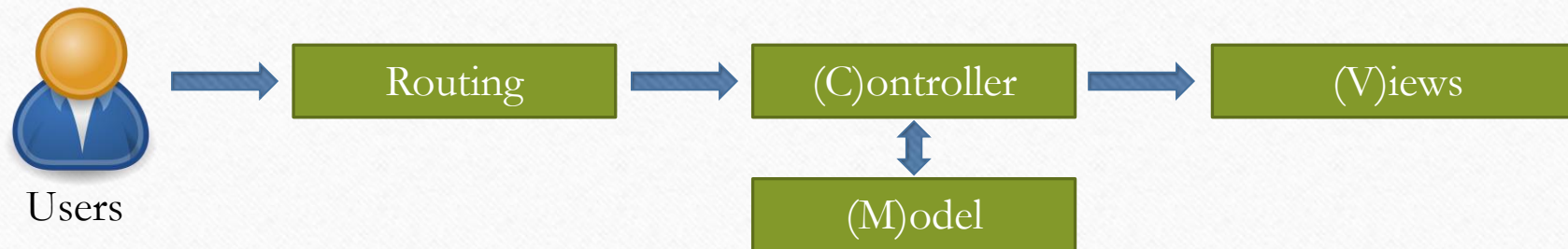
ชั่วโมงที่

2

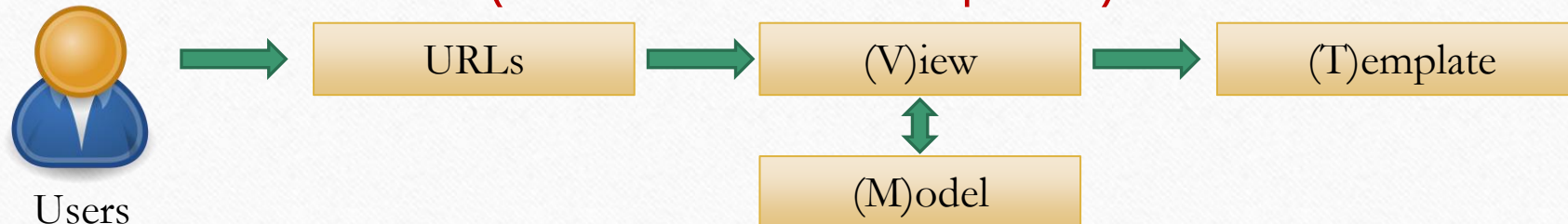
ยูอาร์แอล / วิว & เทมเพลต

MVT

- เฟรมเวิร์คอื่น MVC (Model-View-Controller)



- เฟรมเวิร์คของเรา MVT (Model-View-Template)



URLs

- ยูอาร์แอล คือ ส่วนที่รับ request จากเบราว์เซอร์
- ทำหน้าที่กำหนดเส้นทาง (routing) ที่จะส่งให้ views ตัวไหนทำงาน
- ส่งพารามิเตอร์ไปยัง views ได้อีกด้วย
- ยูอาร์แอลจะอยู่ในไฟล์ชื่อ `urls.py`

URLs : Path

สร้างฟังก์ชัน `hello()` เพื่อรองรับข้อมูลจากยูอาร์แอล

```
from django.shortcuts import render  
from django.http import HttpResponse
```

myapp/views.py

```
def hello(request):  
    return HttpResponse('Hello World')
```


URLs : Path (ต่อ)

การเขียนยูอาร์แอลจะใช้คำสั่ง **path** ดังนี้

```
from myapp import views
```

testproject/urls.py

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

เพิ่มบรรทัดนี้ลงไป

URLs : Path (ต่อ)

หลักการทำงาน

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

เพิ่มฟังก์ชัน hello เข้ามา

ส่งไปให้ฟังก์ชัน hello ทำงาน

URLs : Parameter

การส่งพารามิเตอร์

urls.py

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('hello/<int:id>', views.hello),  
]
```

myapp/views.py

```
from django.http import HttpResponse  
  
def hello(request, id):  
    greeting = 'Your Id: {0}'.format(id)  
    return HttpResponse(greeting)
```


URLs : RePath

สามารถใช้ regular expression เพื่อควบคุม **pattern** ของยูอาร์แอลตาม
ต้องการได้

urls.py

```
from django.urls import path, re_path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello/<int:id>', views.hello),
    path('hello/<int:id>/<str:name>', views.hello),
    re_path(r'article/$', views.article),
]
```

ขึ้นต้นด้วย "r"
ปิดท้ายด้วย "\$"

myapp/views.py

```
def article(request):
    result = 'The article page'
    return HttpResponse(result)
```



URLs : RePath (ต่อ)

การส่งพารามิเตอร์ด้วย regular expression

```
urlpatterns = [  
    re_path(r'article/(?P<year>[0-9]{4})/(?P<slug>[\w-]+)/$', views.article),  
]
```

urls.py

ตัวแปร year
[0-9] คือ ตัวเลข 0-9
{4} คือ ขนาด 4 ตัว

ตัวแปร slug
[\w-] คือ สตริงที่มี "-"
+ คือ ความยาวกี่ตัวก็ได้

URLs : Name

เราสามารถตั้งชื่อได้ โดยใช้คำสั่ง `name=“ชื่อยูอาร์แอล”`

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('hello/<int:id>', views.hello, name='hello1'),  
    path('hello/<int:id>/<str:name>', views.hello, name='hello2'),  
    re_path(r'article/(?P<year>[0-9]{4})/(?P<slug>[\w-]+)/$', views.article, name='article'),  
]
```

urls.py

ตั้งชื่อให้กับยูอาร์แอล

URLs : Include

เราสามารถสืบทอดยูอาร์แอล เพื่อสะดวกในการจัดการได้ โดยใช้คำสั่ง
include

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

include มาใช้งาน

urls.py

URLs : Namespace

สามารถตั้งชื่อ **namespace** เพื่อเป็นตัวแทนของ “กลุ่ม” ยูอาร์แอลได้

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', include(('myapp.urls', 'myapp'), namespace='myapp')),  
]
```

urls.py

เพิ่มคำสั่งลงไป

เทมเพลต

เทมเพลต คือ หน้าจอ HTML ในการแสดงผล ตั้งค่าเทมเพลตในไฟล์ **settings.py**

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [  
            os.path.join(BASE_DIR, 'templates'),  
        ],  
        'APP_DIRS': True,  
        ...  
    },  
]
```

settings.py

ตั้งชื่อโฟลเดอร์

การส่งข้อมูลไปเทมเพลต

จะส่งโดยใช้ dictionary

myapp/views.py

```
from django.http import HttpResponse

def index(request):
    id = '001'
    name = 'Somchai'
    email = 'somchai@mail.com'

    return render(request, 'index.html', {
        'id': id,
        'name': name,
        'email': email,
    })
```

templates/index.html

```
<body>
  <h1>สวัสดี</h1>
  <h2>ID: {{ id }}</h2>
  <h2>Name: {{ name }}</h2>
  <h2>Email: {{ email }}</h2>
</body>
```

แสดงค่าตัวแปร

ส่งข้อมูลเป็น dict

คำสั่งเทมเพลต (ต่อ)

คำสั่งทางเลือก จะใช้ `{% if เงื่อนไข %}` ... `{% endif %}`

```
{% if id == '001' and name == 'Somchai' %}  
    <h2>เข้าสู่ระบบ</h2>  
{% else %}  
    <h2>ไม่สามารถเข้าสู่ระบบได้</h2>  
{% endif %}
```

templates/index.html



คำสั่งเทมเพลต (ต่อ)

คำสั่งวนรอบ จะใช้ `{% for ตัวแปร in อารีย์ %}` ... `{% endfor %}`

myapp/views.py

```
def index(request):
    activities = ['Football', 'Running', 'Badminton']

    return render(request, 'index.html', {
        'activities': activities,
    })
```

templates/index.html

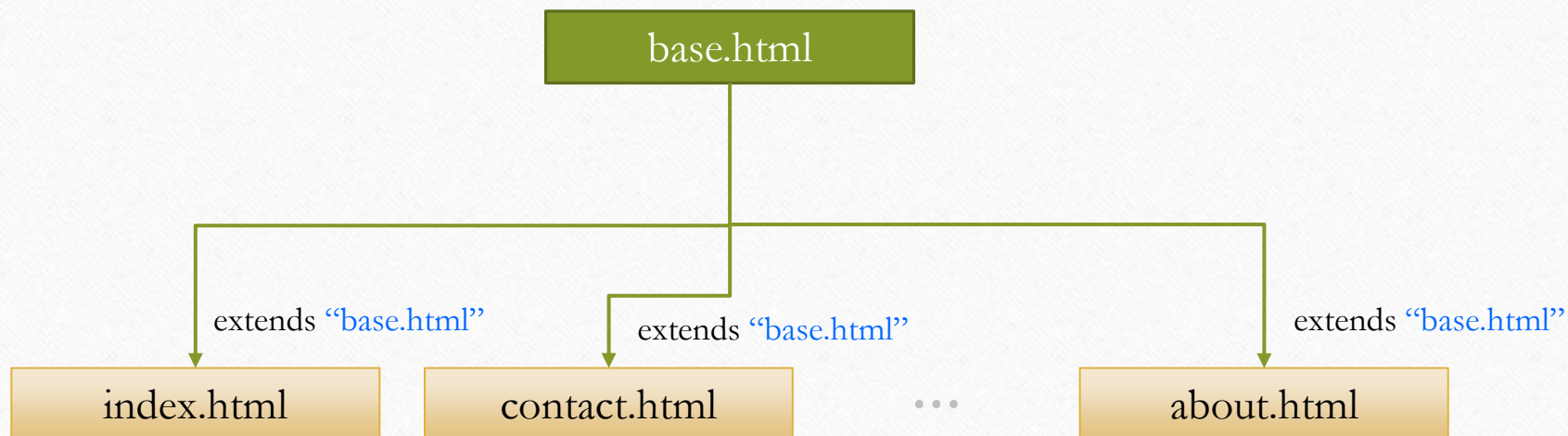
```
<ul>
    {% for a in activities %}
    <li>{{ a }}</li>
    {% endfor %}
</ul>
```


คำสั่งแทมเพลต (ต่อ)

คำสั่งลิงค์ไปยังอาร์แอลอื่นๆ จะใช้ `{% url "namespace:url name" %}`

การสืบทอดเทมเพลต

จะใช้คำสั่ง `{% extends %}` และ `{% block %}` ดังนี้



ชั่วโมงที่

3

โมเดล

รู้จักกับโมเดล

- โมเดล คือ การจำลองข้อมูลในฐานข้อมูล ให้อยู่ในรูปของ “ออบเจกต์” และความสัมพันธ์
- ง่ายในการใช้งาน
- เป็นอิสระต่อฐานข้อมูล

การสร้างโมเดล

```
class Book(models.Model):  
    code = models.CharField(max_length=10, unique=True)  
    slug = models.SlugField(max_length=200, unique=True)  
    name = models.CharField(max_length=100)  
    description = models.TextField(null=True, blank=True)  
    price = models.FloatField(default=0)  
    published = models.BooleanField(default=False)  
    created = models.DateTimeField(auto_now_add=True)  
    updated = models.DateTimeField(auto_now=True)
```

book/models.py

```
def __str__(self):  
    return self.name
```

ฟิลด์ที่แสดงผล เมื่อ
เรียกใช้ object

การสร้างโมเดล (ต่อ)

<https://docs.djangoproject.com/en/3.0/ref/models/fields/>

ฟิลด์	คำอธิบาย
CharField	สตริง
SlugField	สตริงที่มีตัวคั่น
TextField	ข้อความขนาดใหญ่
FloatField	ตัวเลขทศนิยม
BooleanField	บูลีน (จริง / เท็จ)
DateTimeField	วันที่และเวลา
DateField	วันที่
IntegerField	ตัวเลขจำนวนเต็ม
EmailField	อีเมล

โครงสร้างโมเดล

อปชั่น	คำอธิบาย	ตัวอย่าง
null	ค่าว่างหรือไม่ (ในฐานข้อมูล)	null=True
blank	ค่าว่างหรือไม่ (ในหน้าจอ)	blank=True
max_length	จำนวนตัวอักษรสูงสุด	max_length=100
default	ค่าเริ่มต้น	default=10
auto_now_add	เพิ่มวันที่ลงในฟิลด์อัตโนมัติกรณี insert	auto_now_add=True
auto_now	เพิ่มวันที่ลงในฟิลด์อัตโนมัติกรณี update	auto_now=True
unique	กำหนดให้ฟิลด์ข้อมูลห้ามซ้ำ	unique=True
upload_to	กำหนดพารสำหรับ upload	upload_to="/media/upload"

ไมเกรชั่น

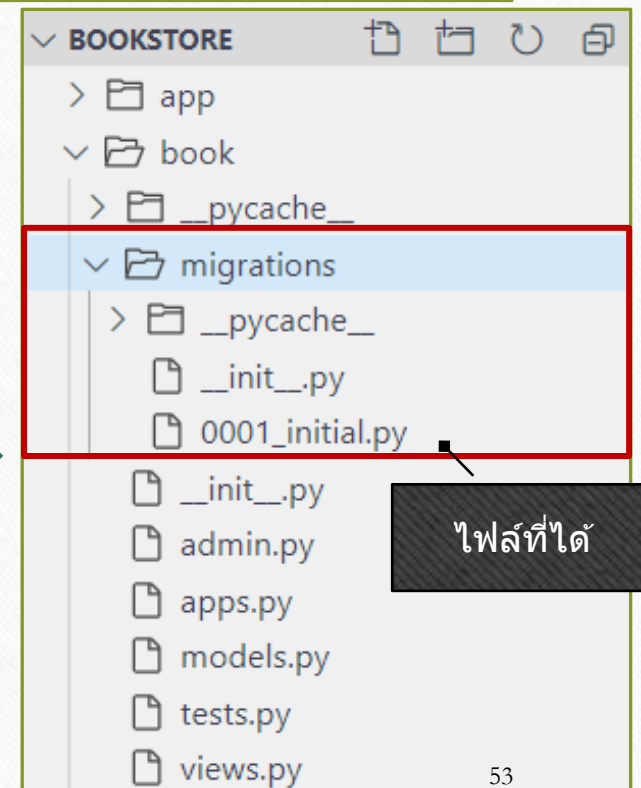
- ไมเกรชั่น (migration) คือ การสร้างสคริปต์เพื่อจัดการฐานข้อมูล
- ไม่ต้องแก้ไขฐานข้อมูลโดยตรง
- มีเวลาตรวจสอบสคริปต์ก่อน
- สามารถ migrate เพื่อเพิ่ม หรือ rollback กรณีผิดพลาดได้
- สะดวกในการติดตั้งบน production

ไมเกรชั่น (ต่อ)

การสร้างไมเกรชั่นใช้คำสั่ง

`python manage.py makemigrations`

```
Command Prompt
(testenv) D:\Training\testenv\bookstore>python manage.py makemigrations
Migrations for 'book':
  book\migrations\0001_initial.py
    - Create model Book
(testenv) D:\Training\testenv\bookstore>_
```



ไมเกรชั่น (ต่อ)

การ apply สคริปต์ลงฐานข้อมูลใช้คำสั่ง `python manage.py migrate`

```
Command Prompt
(testenv) D:\Training\testenv\bookstore>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, book, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying book.0001_initial... OK
  Applying sessions.0001_initial... OK
(testenv) D:\Training\testenv\bookstore>
```

Relationship

- โมเดลสามารถมีความสัมพันธ์ (relationship) กับโมเดลอื่นๆ ได้
- จังโก้มีความสัมพันธ์ให้เราใช้ ดังนี้
 - Many-to-One
 - One-to-One
 - Many-to-Many

Relationship (ต่อ)

ความสัมพันธ์แบบ Many-to-One และ Many-to-Many

หนังสือ 1 เล่มจะอยู่ได้
หมวดหมู่เดียว

Category

1

Author

M

หนังสือ 1 เล่มอาจมีผู้แต่ง
มากกว่า 1 คน

หมวดหมู่ 1 อัน มีหนังสือใน
หมวดนั้นหลายเล่ม

M

Book

M

ผู้แต่ง 1 คนเขียนหนังสือได้
หลายเล่ม



Relationship (ต่อ)

book/models.py

```
class Category(models.Model):  
    name = models.CharField(max_length=100)
```

```
    def __str__(self):  
        return self.name
```

```
class Author(models.Model):  
    name = models.CharField(max_length=100)
```

```
    def __str__(self):  
        return self.name
```

```
class Book(models.Model):  
    category = models.ForeignKey(Category, null=True, blank=True, on_delete=models.CASCADE)  
    author = models.ManyToManyField(Author, blank=True)
```

```
...
```

M-to-1 ใช้คำสั่ง
ForeignKey

Relationship (ต่อ)

ทำการไมเกรทข้อมูลให้เรียบร้อย

```
python manage.py makemigrations
```

```
python manage.py migrate
```

ชั่วโมงที่

4

จัดการข้อมูลด้วย Admin

Admin

ตั้งค่าระบบ admin จะอยู่ในไฟล์ `book/admin.py`

book/admin.py

```
from .models import Category, Author, Book
```

```
admin.site.register(Category)  
admin.site.register(Author)  
admin.site.register(Book)
```

ลงทะเบียนโมเดล

Admin (ต่อ)

สร้าง username และ password สำหรับล็อกอิน

```
python manage.py createsuperuser --username admin --email admin@mail.com
```

Admin Config

ตั้งค่า admin เพิ่มเติม

book/admin.py

```
class BookAdmin(admin.ModelAdmin):  
    list_display = ['code', 'name', 'price', 'category', 'published']  
    list_filter = ['published']  
    search_fields = ['code', 'name']  
  
admin.site.register(Book, BookAdmin)
```

ฟิลด์ที่แสดงผล

ฟิลด์ที่กรองข้อมูล

ฟิลด์ที่ค้นหาข้อมูล

Admin Config (ต่อ)

The screenshot shows the Django Admin interface for a book management system. The browser address bar indicates the URL is `localhost:8000/admin/book/book/`. The page title is "Django administration" and the user is logged in as "ADMIN". A success message states: "The book 'Test Book' was added successfully."

The main content area is titled "Select book to change" and includes a search bar, an "ADD BOOK +" button, and a table of books. The table has columns: CODE, NAME, PRICE, CATEGORY, and PUBLISHED. One book is listed: "Test Book" with CODE "0001" and PRICE "100.0".

Annotations with arrows point to specific parts of the interface:

- search_fields**: Points to the search bar.
- list_display**: Points to the "PRICE" column header in the table.
- list_filter**: Points to the "By published" filter dropdown on the right side of the table.

The bottom left corner features a red logo with the text "ได้ใจ เอ็มเอเอส" (Daijai Emsas).

Admin Config (ต่อ)

book/admin.py

สามารถตั้งค่า slug ได้

```
class BookAdmin(admin.ModelAdmin):  
    list_display = ['code', 'name', 'price', 'category', 'published']  
    list_filter = ['published']  
    search_fields = ['code', 'name']  
    prepopulated_fields = {'slug': ['name']}
```

สร้าง slug จากฟิลด์ name



Admin Config (ต่อ)

สามารถจัดกลุ่มของฟอร์มได้ด้วย **fieldsets**

```
class BookAdmin(admin.ModelAdmin):
```

```
...
```

```
fieldsets = (  
    (None, {'fields': ['code', 'name', 'slug', 'description', 'price']}),  
    ('Category', {'fields': ['category', 'author'], 'classes': ['collapse']}),  
)
```

book/admin.py

Inline

- Inline คือ การจัดการข้อมูลในลักษณะ master – detail ฟอรัม
- มีให้เราใช้หลักๆ 2 แบบ
 - Stacked Inline → ฟอรัมเรียงแถว
 - Tabular Inline → ตาราง

Inline (ต่อ)

เพิ่มโมเดล **BookComment** เพื่อเก็บคอมเมนต์ของผู้ซื้อ

```
class BookComment(models.Model):  
    book = models.ForeignKey(Book, on_delete=models.CASCADE)  
    comment = models.CharField(max_length=100)  
    rating = models.FloatField()  
    created = models.DateTimeField(auto_now_add=True)  
    updated = models.DateTimeField(auto_now=True)  
  
    def __str__(self):  
        return self.name
```

book/models.py

Inline : Stacked Inline

เปิดไฟล์ `admin.py` เพิ่มโค้ดการใช้ StackedInline

```
class BookCommentStackedInline(admin.StackedInline):  
    model = BookComment
```

book/admin.py

สร้างคลาสก่อน

```
class BookAdmin(admin.ModelAdmin):
```

```
    ...  
    inlines = [ BookCommentStackedInline ]
```

นำคลาสมาผูก

```
admin.site.register(Book, BookAdmin)
```


Inline : Tabular Inline (ต่อ)

เปิดไฟล์ `admin.py` เพิ่มโค้ดการใช้ Tabular Inline

```
class BookCommentTabularInline(admin.TabularInline):  
    model = BookComment  
    extra = 2
```

กำหนด extra

```
class BookAdmin(admin.ModelAdmin):  
    ...  
    inlines = [ BookCommentTabularInline ]
```

book/admin.py

อื่นๆ

- การแก้ไขชื่อโมเดล
- การจัดเรียงผลลัพธ์
- ฟิลด์แบบตัวเลือก (choice)

อื่นๆ : การแก้ไขชื่อโมเดล

- จังโก้จะตั้งชื่อโมเดลใน admin ให้เป็น “พหุพจน์”
- เราสามารถเปลี่ยนชื่อตามต้องการได้ โดยใช้ `verbose_name_plural`

```
class Category(models.Model):  
    name = models.CharField(max_length=100)  
  
    class Meta:  
        verbose_name_plural = 'Category'  
  
    def __str__(self):  
        return self.name
```

book/models.py

เพิ่มโค้ดในส่วนนี้



อื่นๆ : การจัดเรียงผลลัพธ์

```
class Book(models.Model):
```

book/models.py

```
...
```

```
class Meta:
```

```
    ordering = ['-created']
```

↓
ติดลบคือ มากไปน้อย

```
    verbose_name_plural = 'Book'
```

```
class BookComment(models.Model):
```

```
...
```

```
class Meta:
```

```
    ordering = ['id']
```

↓
น้อยไปมาก

```
    verbose_name_plural = 'Book Comment'
```



อื่นๆ : การเพิ่มตัวเลือก

เราสามารถเพิ่มตัวเลือกแบบ choice ลงได้

```
BOOK_LEVEL_CHOICE = (  
    ('B', 'Basic'),  
    ('M', 'Medium'),  
    ('A', 'Advance'),  
)
```

ตัวเลือก

book/models.py

```
class Book(models.Model):
```

```
    ...
```

```
    level = models.CharField(max_length=20, choices=BOOK_LEVEL_CHOICE)
```

นำมาใช้



การอัปโหลดไฟล์

เปิดไฟล์ `settings.py` เพิ่มพารามิเตอร์ต่างๆ ดังนี้

settings.py

```
MEDIA_URL = '/media/'
```

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

โฟลเดอร์สำหรับ upload

```
STATIC_URL = '/static/'
```

```
STATIC_ROOT = os.path.join(BASE_DIR, 'static')
```

โฟลเดอร์สำหรับ js, css และ img

```
STATICFILES_DIRS = [  
    os.path.join(BASE_DIR, '/static'),  
]
```


การอัปโหลดไฟล์ (ต่อ)

เปิดไฟล์ `urls.py` เพิ่มการรองรับยูอาร์แอลสำหรับ MEDIA ดังนี้

```
from django.conf import settings
from django.conf.urls.static import static
```

urls.py

```
urlpatterns = [
    path('admin/', admin.site.urls),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

เพิ่มโค้ดชุดนี้

การอัปโหลดไฟล์ (ต่อ)

โมเดลให้ใช้ **FileField** หรือ **ImageField** ก็ได้

- FileField รองรับไฟล์ทุกประเภท
- ImageField รองรับเฉพาะรูปภาพ

```
class Book(models.Model):
```

```
...
```

```
image = models.FileField(upload_to='upload/', null=True, blank=True)
```

กำหนดพาธในการอัปโหลด

book/models.py

การอัปโหลดไฟล์ (ต่อ)

หากต้องการแสดงใน admin ให้เพิ่มโค้ดดังนี้

```
from django.utils.html import format_html
```

book/models.py

```
class Book(models.Model):
```

```
...
```

```
def show_image(self):
```

เพิ่มฟังก์ชัน

```
    if self.image:
```

ส่งออกเป็น HTML

```
        return format_html('' % self.image.url)
```

```
    return ''
```

```
show_image.allow_tags = True
```

บอกระบบให้แปล HTML นั้น

```
show_image.short_description = 'Image'
```



การอัปโหลดไฟล์ (ต่อ)

นำชื่อฟังก์ชันไปใช้ใน `list_display`

book/models.py

```
class BookAdmin(admin.ModelAdmin):  
    list_display = ['code', 'name', 'price', 'category', 'published', 'show_image']  
    ...
```

เพิ่มฟังก์ชัน

ชั่วโมงที่

5

การแสดงรายการ & แบ่งหน้า

Query Set

คำสั่งในการทดสอบ query

```
python manage.py shell
```


Query Set (ต่อ)

การดึงข้อมูลหลายตัว → เป็นอาเรย์

```
books = Book.objects.all()
```

```
books = Book.objects.filter(published=True)
```

เงื่อนไข

```
books = Book.objects.filter(pk__in=[1,2,3])
```

การใช้ like

```
books = Book.objects.filter(name__contains='Test')
```

การใช้ > และ <

```
books = Book.objects.filter(created__gt='2020-01-01', created__lt='2020-12-01')
```

Query Set (ต่อ)

การจัดเรียงผลลัพธ์ใช้ `order_by`

```
books = Book.objects.all().order_by('created')
```

```
books = Book.objects.filter(published=True).order_by('-created', 'name')
```

การจำกัดผลลัพธ์ใช้ [`<index เริ่มต้น base 0>` : `<index สุดท้าย>`]

```
books = Book.objects.all().order_by('created')[2:5]
```

รายการ 3 ถึง 5

```
books = Book.objects.filter(published=True).order_by('-created', 'name')[:10]
```

เอา 10
รายการแรก

Query Set (ต่อ)

การดึงข้อมูลตัวเดียว → เป็นออบเจกต์

```
book = Book.objects.get(pk=1)
```

← ดึงตาม pk

```
book = Book.objects.get(id=1)
```

← ดึงตาม pk

```
book = Book.objects.get(name='Test Book')
```

```
from django.shortcuts import get_object_or_404
```

```
book = get_object_or_404(Book, id=1)
```


แสดงข้อมูลหนังสือ

```
from .models import Category, Book
```

book/views.py

```
def index(request):  
    categories = Category.objects.all()  
    books = Book.objects.filter(published=True)  
  
    return render(request, 'book/index.html', {  
        'books': books,  
        'categories': categories,  
    })
```



แสดงข้อมูลหนังสือ (ต่อ)

ส่ง category id ไปกรองข้อมูล

templates/book/index.html

```
<a href="{% url 'book:index' %}?category={{ c.id }}" class="...">{{ c.name }}</a>
```

```
def index(request):
    categories = Category.objects.all()
    books = Book.objects.filter(published=True)

    category = request.GET.get('category')
    if category:
        books = books.filter(category_id=category)

    return render(request, 'book/index.html', {
        ...
    })
```

กรองข้อมูล

book/views.py

แสดงข้อมูลหนังสือ (ต่อ)

ความสัมพันธ์แบบ m-to-m ให้ต่อท้ายด้วย **all**

templates/book/index.html

```
<strong>Author:</strong>
{% for a in b.author.all %}
<span class="badge badge-secondary">{{ a.name }}</span>
{% endfor %}
```

ต่อท้ายด้วย all



การแบ่งหน้า

ใช้คลาส Paginator

book/views.ph

```
paginator = Paginator(books, 10)
page = request.GET.get('page')
try:
    books = paginator.page(page)
except PageNotAnInteger:
    books = paginator.page(1)
except EmptyPage:
    books = paginator.page(paginator.num_pages)
```

จำนวนรายการ/หน้า

templates/book/index.html

```
<ul class="pagination justify-content-center mb-5">
  {% if books.has_previous %}
  <li class="page-item"><a class="page-link" href="?page={{ books.previous_page_number }}">Previous</a></li>
  {% endif %}
  {% for i in books.paginator.page_range %}
  <li class="page-item {% if books.number == i %}active{% endif %}">
    <a class="page-link" href="?page={{ i }}">{{ i }}</a>
  </li>
  {% endfor %}
  {% if books.has_next %}
  <li class="page-item"><a class="page-link" href="?page={{ books.next_page_number }}">Next</a></li>
  {% endif %}
</ul>
```

แสดงรายละเอียด

```
def detail(request, slug):  
    book = get_object_or_404(Book, slug=slug)  
    return render(request, 'book/detail.html', {  
        'book': book,  
    })
```

ดึงข้อมูลตาม slug

book/views.py

```
urlpatterns = [  
    path('', views.index, name='index'),  
    path('<slug:slug>', views.detail, name='detail'),  
]
```

book/urls.py



การแสดงคอมเมนต์

templates/book/detail.html

```
<div class="card mb-5">
  <div class="card-header">
    <div class="card-title">Comments</div>
  </div>
  <div class="card-body">
    {% for c in book.bookcomment_set.all %}
    <blockquote class="blockquote text-center">
      <p class="mb-0">{{ c.comment }}</p>
      <footer class="small">Rating: {{ c.rating }}</footer>
    </blockquote>
    {% endfor %}
  </div>
</div>
```

หัวใจสำคัญ



การแสดงคอมเมนต์ (ต่อ)

```
class BookComment(models.Model):
    book = models.ForeignKey(Book, on_delete=models.CASCADE, related_name='book_comment')
    comment = models.CharField(max_length=100)
    rating = models.FloatField()
    created = models.DateTimeField(auto_now_add=True)
    updated = models.DateTimeField(auto_now=True)
```

ตั้งชื่อใหม่

```
{% for c in book.book_comment.all %}
<blockquote class="blockquote text-center">
    ...
</blockquote>
{% endfor %}
```

templates/book/detail.html

ใช้ชื่อที่ตั้ง

การแสดงผลคอมเมนต์ (ต่อ)

การนับ comment ให้เพิ่มฟังก์ชันลงในคลาส Book

book/models.py

```
class Book(models.Model):
```

```
...
```

```
def get_comment_count(self):  
    return self.book_comment.count()
```

ฟังก์ชันของเรา

book/detail.html

```
<div class="card-header">  
    <div class="...">  
        Comments ({{ book.get_comment_count }})  
    </div>  
</div>
```

เรียกใช้ฟังก์ชัน

Template Tags

เราสามารถโหลดฟังก์ชันที่เกี่ยวข้องกับ **template** มาใช้งานได้

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'django.contrib.humanize',  
    'app',  
    'book',  
]
```

settings.py

เพิ่มชื่อนี้

```
{% extends "layouts/base.html" %}  
{% load humanize %}  
  
{% block content %}  
  
...  
  
{% endblock %}
```

โหลดใช้งาน

Template Tags (ต่อ)

การใช้งาน

templates/book/detail.html

```
<div class="mt-4 mb-4">{{ book.description }}</div>
<h5 class="mb-4">฿ {{ book.price|floatformat:2|intcomma }}</h5>
<div class="form-inline">
  <input type="number" class="form-control" value="1">
  <a href="#" class="btn btn-success ml-2">Add to Cart</a>
</div>
```

ฟังก์ชันใน humanize

ชั่วโมงที่

6

ฟอร์ม

สร้างฟอร์ม

สร้างไฟล์ `book/forms.py` ดังนี้

book/forms.py

```
from django import forms
from .models import Book
```

```
class BookForm(forms.ModelForm):
```

```
    class Meta:
```

```
        model = Book
```

โมเดลที่จะทำฟอร์ม

```
        exclude = ['id', 'slug', 'published', 'created', 'updated']
```

กรอกรางฟิลด์ออก



สร้างฟอร์ม (ต่อ)

```
from .forms import BookForm
```

book/views.py

```
def book_add(request):
```

```
    form = BookForm()
```

ฟอร์ม

```
    return render(request, 'book/add.html', {  
        'form': form,  
    })
```

ส่งไปหน้าจอ

```
urlpatterns = [
```

```
    path('', views.index, name='index'),
```

```
    path('detail/<slug:slug>/', views.detail, name='detail'),
```

```
    re_path(r'^add/$', views.book_add, name='book_add'),
```

```
    re_path(r'^edit/(?P<slug>[\w-]+)/$', views.book_edit, name='book_edit'),
```

```
]
```

ผูกยูอาร์แอล

book/urls.py

สร้างฟอร์ม (ต่อ)

แสดงฟอร์มในหน้าจอ

templates/book/add.html

```
<h1>Add Book</h1>
<form method="post" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form }}
    <button type="submit" class="btn btn-success">Save Book</button>
</form>
```

รองรับการ upload

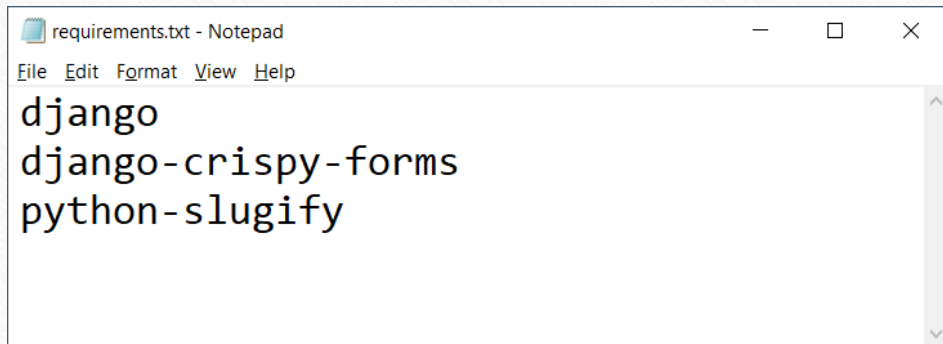
ป้องกัน CSRF

ฟอร์มของเรา

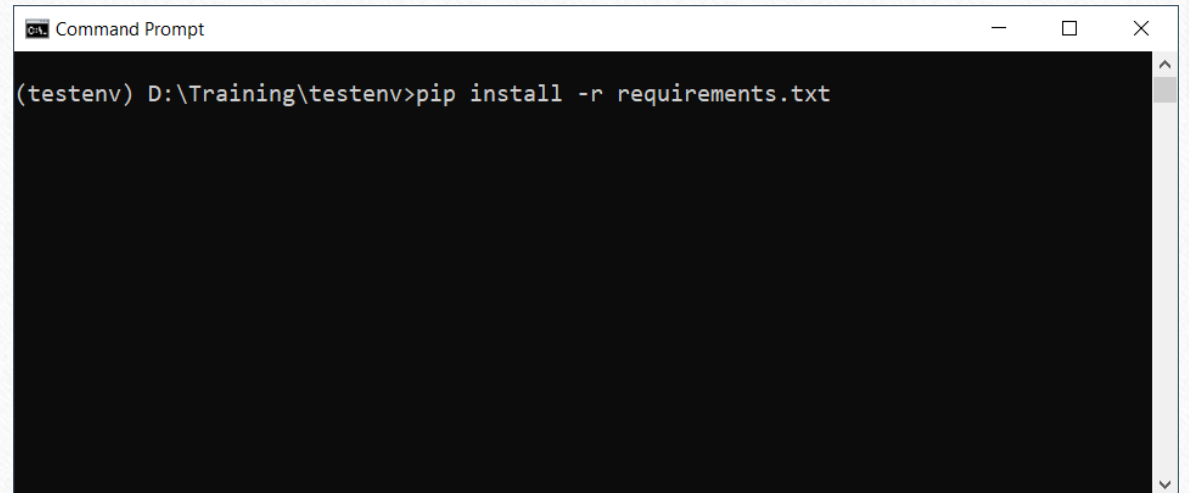


สร้างฟอร์ม (ต่อ)

ติดตั้ง **crispy-forms** เพื่อรองรับ Bootstrap 4



```
requirements.txt - Notepad
File Edit Format View Help
django
django-crispy-forms
python-slugify
```



```
Command Prompt
(testenv) D:\Training\testenv>pip install -r requirements.txt
```


สร้างฟอร์ม (ต่อ)

โหลด **crispy-forms** มาใช้งาน

```
{% load humanize %}
{% load crispy_forms_tags %}

{% block content %}
<form method="post" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form|crispy }}
    <button type="submit" class="btn btn-success">Save Book</button>
</form>
{% endblock %}
```

templates/book/add.html



สร้างฟอร์ม (ต่อ)

```
def book_add(request):  
    form = BookForm()
```

book/views.py

```
    if request.method == 'POST':
```

```
        form = BookForm(request.POST, request.FILES)
```

อ่านข้อมูลจากฟอร์ม

```
        if form.is_valid():
```

```
            book = form.save(commit=False)
```

แปลงข้อมูลเป็นออบเจกต์

```
            book.slug = slugify(book.name)
```

```
            book.published = True
```

```
            book.save()
```

```
            form.save_m2m()
```

```
            messages.success(request, 'Save success')
```

```
            return HttpResponseRedirect(reverse('book:index', kwargs={}))
```

redirect กลับไปหน้าแรก

```
            messages.error(request, 'Save Failed!')
```

```
    return render(request, 'book/add.html', {
```

```
        'form': form,
```

```
    })
```

สร้างฟอร์ม (ต่อ)

แสดง message ที่หน้าจอ

```
{% if messages %}
  {% for message in messages %}
    {% ifequal message.tags 'success' %}
      <div class="alert alert-success alert-dismissible fade show" role="alert">
        <strong>{{ message }}</strong>
        <button type="button" class="close" data-dismiss="alert" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
    {% endifequal %}
  {% endfor %}
{% endif %}
```

templates/layouts/base.html

Custom Validation

```
class BookForm(forms.ModelForm):
    class Meta:
        model = Book
        exclude = ['id', 'slug', 'published', 'created', 'updated']

    def __init__(self, *args, **kwargs):
        super(BookForm, self).__init__(*args, **kwargs)
        self.fields['code'].error_messages = {
            'required': 'Please enter book code',
        }
        self.fields['name'].error_messages = {
            'required': 'Please enter book name',
        }
        self.fields['price'].error_messages = {
            'required': 'Please enter book price',
            'invalid': 'Please enter a valid book price',
        }
```

book/forms.py

Override ฟังก์ชันเพื่อ
ตรวจสอบ field

Custom Validation (ต่อ)

book/forms.py

```
class BookForm(forms.ModelForm):
    class Meta:
        model = Book
        exclude = ['id', 'slug', 'published', 'created', 'updated']

    def __init__(self, *args, **kwargs):
        super(BookForm, self).__init__(*args, **kwargs)
        ...

    def clean(self):
        cleaned_data = super(BookForm, self).clean()
        if not cleaned_data.get('category'):
            self.add_error('category', 'Please select category name')

        if not cleaned_data.get('author'):
            self.add_error('author', 'Please select author name')
```

override ฟังก์ชัน clean

ชั่วโมงที่

7

คลาสเบสวิว

เกี่ยวกับ CBV

- จังโก้จะมีการเขียน view สองแบบ
 - **Function-based View** → ที่เรามาแล้ว
 - **Class-based View** → แนวทางใหม่
- Class-based View (CBV) ช่วยลดความซ้ำซ้อนของโค้ด

CBV มีอะไรให้ใช้บ้าง?

ที่ใช้บ่อยๆ มีดังนี้

- ListView
- DetailView
- CreateView
- UpdateView
- DeleteView

CBV : ListView

book/views.py

```
from django.views.generic import ListView
```

```
class BookListView(ListView):  
    model = Book  
    template_name = 'book/index.html'  
    context_object_name = 'books'  
    paginate_by = 5  
  
    def get_queryset(self):  
        return Book.objects.filter(published=True)  
  
    def get_context_data(self, *args, **kwargs):  
        context = super(BookListView, self).get_context_data(*args, **kwargs)  
        context.update({  
            'categories': Category.objects.all(),  
        })  
        return context
```

โมเดล

QuerySet

ข้อมูลอื่นๆ ที่ส่งไปด้วย

CBV : ListView (ต่อ)

เปลี่ยนยูอาร์แอลใหม่

book/urls.py

```
from . import views
```

```
urlpatterns = [
```

เปลี่ยนใหม่

```
    path('', views.BookListView.as_view(), name='index'),  
    path('detail/<slug:slug>', views.detail, name='detail'),  
    re_path(r'add/$', views.book_add, name='book_add'),  
    re_path(r'edit/(?P<slug>[\w-]+)/$', views.book_edit, name='book_edit'),
```

```
]
```

CBV : Detail View

การแสดงผลละเอียดของหนังสือ

book/views.py

```
class BookDetailView(DetailView):  
    model = Book  
    template_name = 'book/detail.html'  
    slug_url_kwarg = 'slug'
```

CBV : Detail View (ต่อ)

เปลี่ยนยูอาร์แอลใหม่

เปลี่ยนใหม่

book/urls.py

```
urlpatterns = [  
    path('', views.BookListView.as_view(), name='index'),  
    path('detail/<slug:slug>', views.BookDetailView.as_view(), name='detail'),  
    re_path(r'add/$', views.book_add, name='book_add'),  
    re_path(r'edit/(?P<slug>[\w-]+)/$', views.book_edit, name='book_edit'),  
]
```



ชั่วโมงที่

8

Sessions

การใช้ Session (ต่อ)

ตัวอย่างการใช้งาน

```
request.session['my_car'] = 'mini'
```

เพิ่มข้อมูล

```
my_car = request.session['my_car']
```

ดึงข้อมูล

```
my_car = request.session.get('my_car', 'mini')
```

ดึงข้อมูลแบบมีค่า default

```
del request.session['my_car']
```

ลบข้อมูล

ตะกร้าสินค้า

book/views.py

```
def cart_add(request, slug):
    book = get_object_or_404(Book, slug=slug)
    cart_items = request.session.get('cart_items') or []
    book_name = '[Book] ' + book.name

    duplicated = False
    for c in cart_items:
        if c.get('name') == book_name:
            c['qty'] = int(c.get('qty') or '1') + 1
            duplicated = True

    if not duplicated:
        cart_items.append({
            'id': book.id, 'name': book_name, 'slug': book.slug, 'price': book.price, 'qty': 1,
        })
    request.session['cart_items'] = cart_items
    return HttpResponseRedirect(reverse('book:cart_list', kwargs={}))
```

ดึงมาจาก session

ถ้ามีสินค้าอยู่แล้ว ให้
เพิ่มจำนวนอีก 1

ถ้าไม่มีสินค้า ให้เพิ่ม
สินค้าใหม่ลงไป

บันทึกกลับลงใน session

ตะกร้าสินค้า (ต่อ)

แสดงสินค้าในตะกร้า

book/views.py

```
def cart_list(request):  
    cart_items = request.session.get('cart_items') or []  
    total_qty = 0  
    for c in cart_items:  
        total_qty += c.get('qty')  
    request.session['cart_qty'] = total_qty  
    return render(request, 'book/cart.html', {  
        'menu': 'book',  
    })
```


ดึงมาจาก session

รวมจำนวนทั้งหมด

ตะกร้าสินค้า (ต่อ)

ลบสินค้าออกจากตะกร้า

book/views.py

```
def cart_delete(request, slug):  
    cart_items = request.session.get('cart_items') or []  
    for i in range(len(cart_items)):  
        if cart_items[i]['slug'] == slug:   
            del cart_items[i]  
            break  
    request.session['cart_items'] = cart_items  
    return HttpResponseRedirect(reverse('book:cart_list', kwargs={}))
```

ลบสินค้าที่เลือก ออก
จาก session

ตะกร้าสินค้า (ต่อ)

ยูอาร์แอล

book/urls.py

```
urlpatterns = [  
    path('', views.BookListView.as_view(), name='index'),  
    path('detail/<slug:slug>/', views.BookDetailView.as_view(), name='detail'),  
    re_path(r'add/$', views.book_add, name='book_add'),  
    re_path(r'edit/(?P<slug>[\w-]+)/$', views.book_edit, name='book_edit'),  
  
    re_path(r'^cart/list/$', views.cart_list, name='cart_list'),  
    re_path(r'^cart/add/(?P<slug>[\w-]+)$', views.cart_add, name='cart_add'),  
    re_path(r'^cart/delete/(?P<slug>[\w-]+)$', views.cart_delete, name='cart_delete'),  
]
```



ตะกร้าสินค้า (ต่อ)

หน้าจอบ่งแสดงจำนวนสินค้า (มุมมองเว็บ)

layouts/base.html

```
{% if request.session.cart_qty %}  
<a href="{% url 'book:cart_list' %}" class="btn btn-outline-secondary mx-2">  
  <i class="fa fa-shopping-cart"></i>  
  <span>{{ request.session.cart_qty }}</span>  
</a>  
{% endif %}
```

ตรวจสอบว่ามีสินค้า
ในตะกร้าหรือไม่?

แสดงจำนวนสินค้า

ตะกร้าสินค้า (ต่อ)

book/cart.html

```
<table class="table table-bordered table-hover table-cart">
  <thead>
    <tr>
      <th>Items</th>
      <th>Qty</th>
      <th style="text-align:right" width="15%">Unit Price</th>
      <th style="text-align:center"></th>
    </tr>
  </thead>
  <tbody>
    {% for c in request.session.cart_items %}
    <tr>
      <td>{{ c.name }}</td>
      <td><input type="text" class="form-control" value="{{ c.qty }}"></td>
      <td>{{ c.price|floatformat:2|intcomma }}</td>
      <td width="80px"><a href="{% url 'book:cart_delete' slug=c.slug %}"> Delete</a></td>
    </tr>
    {% endfor %}
  </tbody>
</table>
```

▼
วนลูปสินค้าจากตะกร้า

ชั่วโมงที่

9

ล็อกอิน / ล็อกเอาต์

Login

```
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth import login, logout
```

app/views.py

```
def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('book:index')
    else:
        form = AuthenticationForm()
    return render(request, 'account/login.html', {'form': form})
```

Login (ต่อ)

หน้าจอล็อกอิน

```
{% load crispy_forms_tags %}
```

templates/account/login.html

```
{% block content %}
```

```
<div class="container">
```

```
<h1>Log in</h1>
```

```
<form method="post" novalidate>
```

```
    {% csrf_token %}
```

```
    {{ form|crispy }}
```

```
    <button type="submit" class="btn btn-success">Log in</button>
```

```
</form>
```

```
</div>
```

```
{% endblock %}
```

Login (ต่อ)

เพิ่มยูอาร์แอล

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

app/urls.py

```
urlpatterns = [  
    path('', views.index, name='index'),  
    path('login/', views.login_view, name='login'),  
    path('logout/', views.logout_view, name='logout'),  
    path('signup/', views.signup_view, name='signup'),  
]
```


Login (ต่อ)

ตรวจสอบ Login

```
def login_view(request):  
    if request.method == 'POST':  
        form = AuthenticationForm(data=request.POST)  
        if form.is_valid():  
            user = form.get_user()  
            login(request, user)  
            return redirect('book:index')  
    else:  
        form = AuthenticationForm()  
    return render(request, 'account/login.html', {'form': form})
```

app/views.py

ตรวจสอบฟอร์ม และ
เรียกใช้ฟังก์ชัน login



Log out

สร้างฟังก์ชันล็อกเอาท์

```
def logout_view(request):  
    if request.method == 'POST':  
        logout(request)  
        return redirect('book:index')
```

app/views.py

```
urlpatterns = [  
    path('', views.index, name='index'),  
    path('login/', views.login_view, name='login'),  
    path('logout/', views.logout_view, name='logout'),  
    path('signup/', views.signup_view, name='signup'),  
]
```

app/urls.py

Log out (ต่อ)

เพิ่มปุ่ม Log out ที่หน้าจอ

ตรวจสอบว่า login หรือไม่

แสดงชื่อ user

templates/layouts/base.html

```
{% if user.is_authenticated %}
<span>Hello, {{request.user.username}}</span>
<form action="{% url 'app:logout' %}" method="post">
  {% csrf_token %}
  <button class="btn btn-danger ml-2" type="submit">Log Out</button>
</form>
{% endif %}
```


Sign up

สร้างวิวสำหรับฟอร์มลงทะเบียน

```
def signup_view(request):  
    if request.method == 'POST':  
        form = UserCreationForm(request.POST)  
        if form.is_valid():  
            user = form.save()  
            login(request, user)  
            return redirect('book:index')  
    else:  
        form = UserCreationForm()  
    return render(request, 'account/signup.html', {'form': form})
```

app/views.py

Sign up (ต่อ)

ผูกยูอาร์แอล

```
urlpatterns = [  
    path('', views.index, name='index'),  
    path('login/', views.login_view, name='login'),  
    path('logout/', views.logout_view, name='logout'),  
    path('signup/', views.signup_view, name='signup'),  
]
```

app/urls.py

Sign up (ต่อ)

ฟอร์มสำหรับลงทะเบียน

```
{% load crispy_forms_tags %}

{% block content %}
<div class="container">
  <h1>Sign up</h1>
  <form method="post" novalidate>
    {% csrf_token %}
    {{ form|crispy }}
    <button type="submit" class="btn btn-success">Register User</button>
  </form>
</div>
{% endblock %}
```

templates/account/signup.html

การตรวจสอบสิทธิ์

การซ่อนปุ่มหรือเมนู จะใช้ `is_authenticated`

แสดงปุ่ม Sign up และ Log in
หากยังไม่ได้เข้าระบบ

templates/layouts/base.html

```
{% if not user.is_authenticated %}  
<a class="btn btn-outline-warning ml-2" href="{% url 'app:signup' %}">Sign up</a>  
<a class="btn btn-outline-primary ml-2" href="{% url 'app:login' %}">Log in</a>  
{% endif %}
```

การตรวจสอบสิทธิ์ (ต่อ)

การป้องกันการเข้าถึง โดยไม่ได้ล็อกอินจะใช้ decorator ชื่อ **login_required**

```
from django.contrib.auth.decorators import login_required
```

app/urls.py

```
urlpatterns = [  
    path('', login_required(vIEWS.BookListView.as_view()), login_url='/login'), name='index'),  
    path('detail/<slug:slug>/', views.BookDetailView.as_view(), name='detail'),  
    re_path(r'add/$', views.book_add, name='book_add'),  
    re_path(r'edit/(?P<slug>[\w-]+)/$', views.book_edit, name='book_edit'),  
]
```

ป้องกันการเข้าถึง และให้
redirect ไป login ใหม่



ชั่วโมงที่

10

อีเมลล์

ตั้งค่าอีเมล

เปิดไฟล์ `settings.py` ตั้งค่าอีเมลต่อไปนี้

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'  
EMAIL_USE_TLS = True  
EMAIL_HOST = 'smtp.gmail.com'  
EMAIL_PORT = 587  
EMAIL_HOST_USER = 'xxx@gmail.com'  
EMAIL_HOST_PASSWORD = 'xxxxxxxxxx'
```

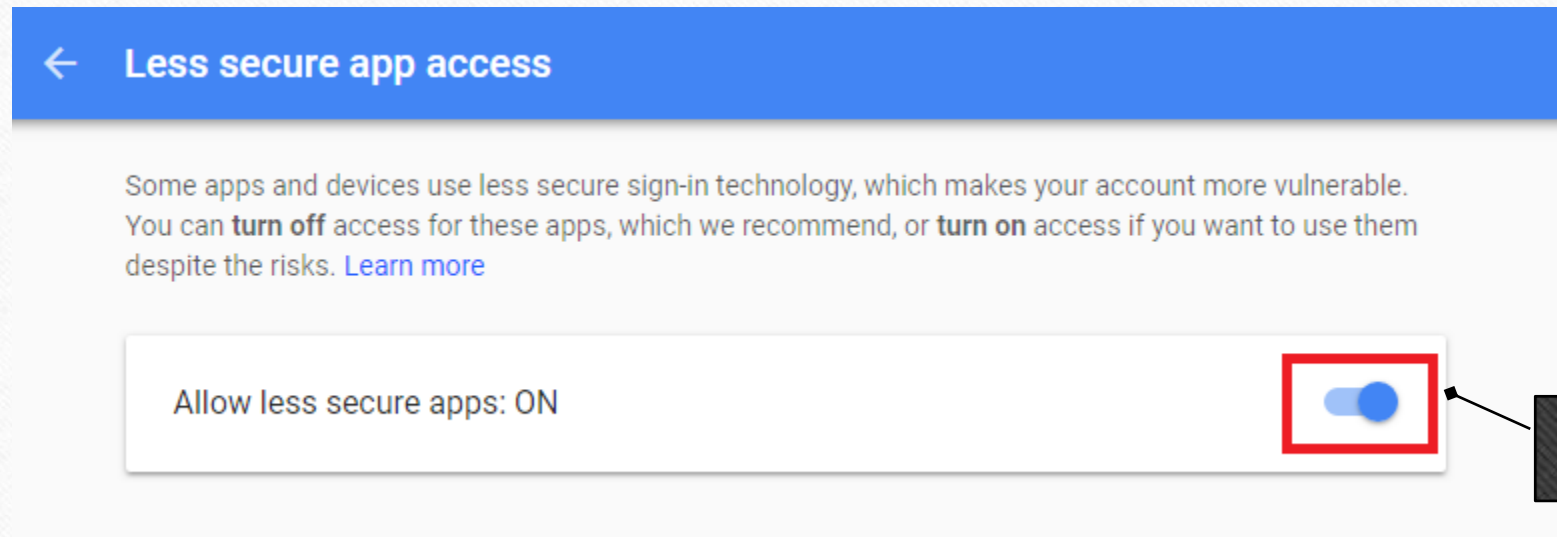
settings.py

ตั้งค่า SMTP Server



ตั้งค่าอีเมล (ต่อ)

เปิดใช้ Less Secure App



ส่งอีเมล

เขียนโค้ดเพื่อส่งอีเมล

book/views.py

```
from django.core.mail import EmailMessage

def cart_checkout(self):
    subject = 'Test Email'
    body = '''
        <p>This is a test mail message</p>
    '''
    email = EmailMessage(subject=subject, body=body, to=['info@xxx.com'])
    email.content_subtype = 'html'
    email.send()

    return HttpResponseRedirect(reverse('book:index', kwargs={}))
```

รูปแบบข้อความ

ส่งอีเมล (ต่อ)

ยูอาร์แอล

book/urls.py

```
urlpatterns = [  
    re_path(r'^cart/list/$', views.cart_list, name='cart_list'),  
    re_path(r'^cart/add/(?P<slug>[\w-]+)$', views.cart_add, name='cart_add'),  
    re_path(r'^cart/delete/(?P<slug>[\w-]+)$', views.cart_delete, name='cart_delete'),  
    re_path(r'^cart/checkout/$', views.cart_checkout, name='cart_checkout'),  
]
```

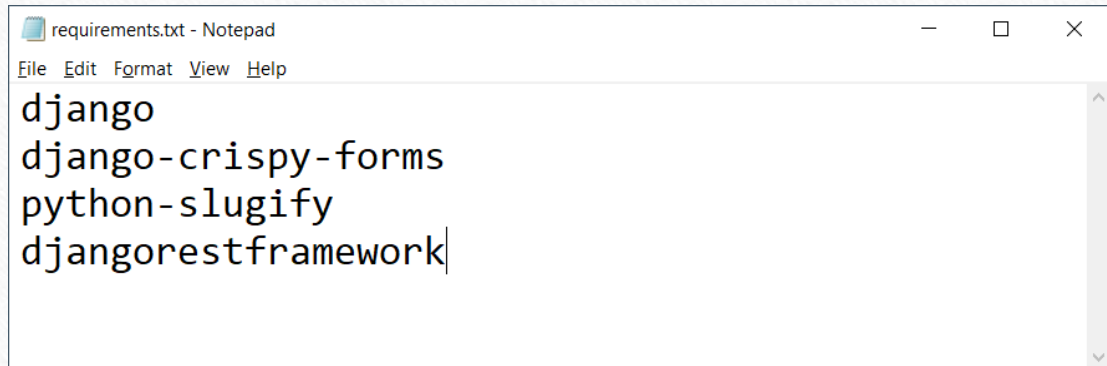
ชั่วโมงที่

11

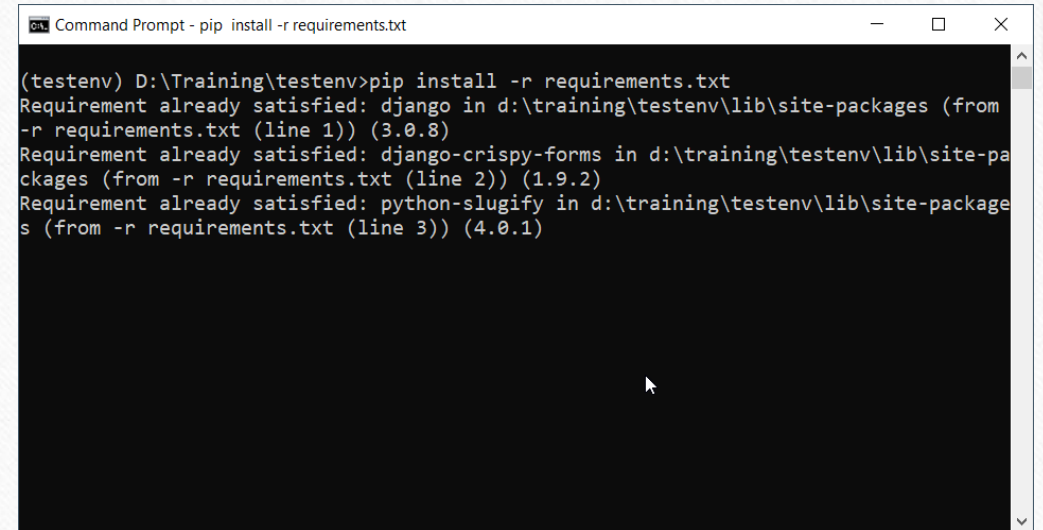
การทำ Rest API

การติดตั้ง

- ไลบรารีทำ Rest API ที่ได้รับความนิยมชื่อว่า **djangoRESTframework**
- เพิ่มลงใน **requirements.txt** และติดตั้ง



```
requirements.txt - Notepad
File Edit Format View Help
django
django-crispy-forms
python-slugify
djangoRESTframework
```



```
Command Prompt - pip install -r requirements.txt
(testenv) D:\Training\testenv>pip install -r requirements.txt
Requirement already satisfied: django in d:\training\testenv\lib\site-packages (from
-r requirements.txt (line 1)) (3.0.8)
Requirement already satisfied: django-crispy-forms in d:\training\testenv\lib\site-pa
ckages (from -r requirements.txt (line 2)) (1.9.2)
Requirement already satisfied: python-slugify in d:\training\testenv\lib\site-package
s (from -r requirements.txt (line 3)) (4.0.1)
```


การติดตั้ง (ต่อ)

เพิ่มลงใน INSTALL_APPS

```
INSTALLED_APPS = [  
    ...  
    'rest_framework',  
]
```

Serializers

Serializer คือ คลาสที่ใช้ในการ “ห่อ” (serialized) หรือ “แกะห่อ” (de-serialized) ข้อมูลที่อยู่ใน object

book/serializers.py

```
from rest_framework import serializers
from .models import Book
```

```
class BookSerializer(serializers.ModelSerializer):
```

```
    class Meta:
```

```
        model = Book
```

```
        fields = ['code', 'name', 'description', 'price']
```

ชื่อโมเดล

ฟิลด์ที่ต้องการ



API View

book/api.py

```
from rest_framework import routers, serializers, viewsets
from .models import Book
from .serializers import BookSerializer
```

```
class BookViewSet(viewsets.ModelViewSet):
    queryset = Book.objects.all()
    serializer_class = BookSerializer
```

สร้างวิว

กำหนดตัว serializer

```
router = routers.DefaultRouter()
router.register(r'book/list', BookViewSet)
```

กำหนด URL ลงใน router



API View (ต่อ)

ผูก router เข้ากับยูอาร์แอลหลัก

```
from book.api import router
```

urls.py

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', include(('app.urls', 'app'), namespace='app')),  
    path('book/', include(('book.urls', 'book'), namespace='book')),  
    path('api/', include(router.urls)),  
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

ผูกเข้าด้วยกัน

Nested Serializers

book/serializers.py

```
class CategorySerializer(serializers.ModelSerializer):  
    class Meta:  
        model = Category  
        fields = ['id', 'name']
```

```
class AuthorSerializer(serializers.ModelSerializer):  
    class Meta:  
        model = Author  
        fields = ['id', 'name']
```

```
class BookSerializer(serializers.ModelSerializer):  
    category = CategorySerializer(read_only=True)  
    author = AuthorSerializer(many=True, read_only=True)  
    class Meta:  
        model = Book  
        fields = ['code', 'name', 'description', 'category', 'author', 'price']
```

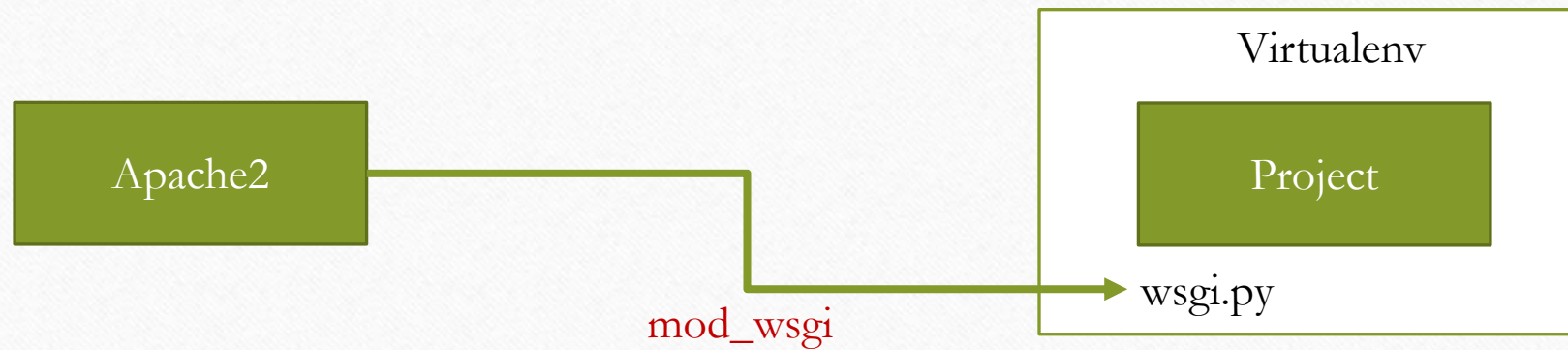
ใช้ many=True กรณี
ออกมาหลายตัว

ชั่วโมงที่

12

การติดตั้งลงบนเซิร์ฟเวอร์

ภาพรวม



การติดตั้ง

- ติดตั้งฐานข้อมูล
- ติดตั้ง **virtualenv** บนคลาวด์
- นำโค้ดโปรเจกต์ของเรา ไปรันใน **virtualenv**
- ติดตั้ง Apache2
- ติดตั้ง **mod_wsgi** บน Apache2
- ติดตั้ง Let's encrypt บน Apache2 เพื่อรองรับ SSL (กฤษฎาเขียว)

การติดตั้ง (ต่อ)

เอกสารคู่มือการติดตั้ง

https://www.digitalocean.com/community/tutorials/how-to-serve-django-applications-with-apache-and-mod_wsgi-on-ubuntu-16-04

ติดตั้ง Let's encrypt

<https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-ubuntu-18-04>