Build your portfolio site with Django

Getting started with Python’s most famous web framework

- Part 1 - Django Overview
- Part 2 - Set up your Development Environment
- Part 3 - Build a Django application
- Part 4 - Showcase your Projects
- Part 5 - Display a single Project

PART 1 - DJANGO OVERVIEW

PART 1 - DJANGO OVERVIEW

▶ **1. About this section**

2. About Web Frameworks and Django
3. What you'll build and learn
4. Apps in a Django project
5. Files in a Django project
6. Flow in a Django project
7. Recap and Outlook

Relax and enjoy the ride!

Relax and enjoy the ride!

PART 1 - DJANGO OVERVIEW

1. About this section
2. **About Web Frameworks and Django**
3. What you'll build and learn
4. Apps in a Django project
5. Files in a Django project
6. Flow in a Django project
7. Recap and Outlook

Why use a web framework?

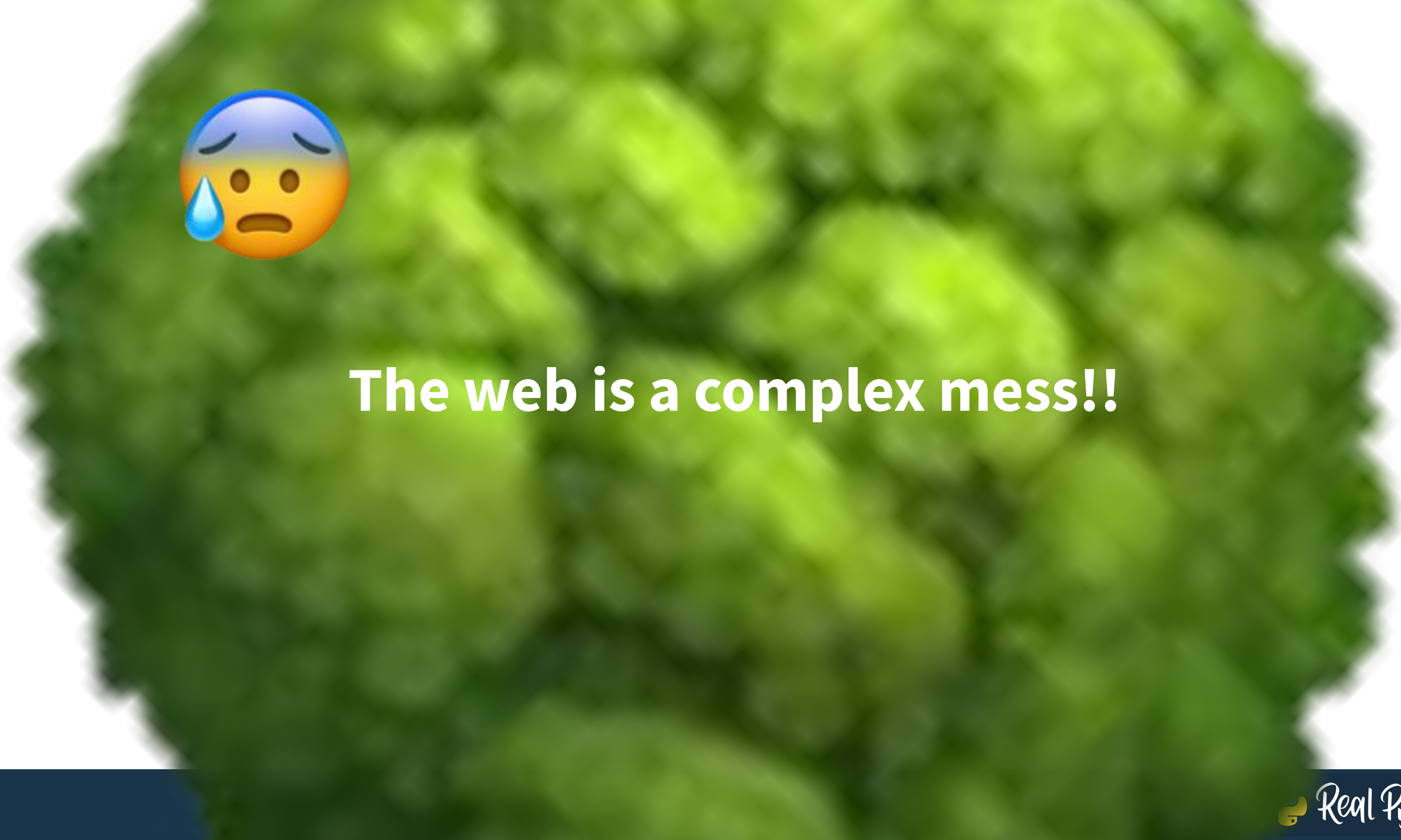
The web grew organically

The web 🌱 grew organically

🌳

The web grew organically

The web grew organically



The web is a complex mess!!

Web frameworks reduce complexity

Web frameworks

- Automate repetitive web development tasks
- Abstract database interactions
- Handle URL requests and URL mapping
- Provide templating frameworks
- Help with security, caching, sessions…
- Generally make your life easier :)

django

Django is a Python web framework

Why Django?

- Fantastic community
- Time- and battle-tested
- Batteries included!
- Huge third-party package ecosystem
- Pluggable structure - reuse apps
- Django ORM for database interactions
- And… It’s in Python! :)

PART 1 - DJANGO OVERVIEW

1. About this section
2. About Web Frameworks and Django
3. **What you’ll build and learn**
4. Apps in a Django project
5. Files in a Django project
6. Flow in a Django project
7. Recap and Outlook

Your portfolio as a webapp!

What you’ll learn in this course

- ✓ How to build a Django project from start to finish

What you’ll learn in this course

- ✓ How to build a Django project from start to finish
- ✓ We’ll do lot’s of Django deep-dives, asking: *how does this actually work?*

What you’ll learn in this course

- ✓ How to build a Django project from start to finish
- ✓ We’ll do lot’s of Django deep-dives, asking: *how does this actually work?*
- ✓ Get friendly with Django error messages

What you’ll learn in this course

- ✓ How to build a Django project from start to finish
- ✓ We’ll do lot’s of Django deep-dives, asking: *how does this actually work?*
- ✓ Get friendly with Django error messages
- ✓ Hands-on debugging approaches during Django development

What you’ll learn in this course

- ✓ How to build a Django project from start to finish
- ✓ We’ll do lot’s of Django deep-dives, asking: *how does this actually work?*
- ✓ Get friendly with Django error messages
- ✓ Hands-on debugging approaches during Django development
- ✓ Understand Django projects and file structure

What you’ll learn in this course

- ✓ How to build a Django project from start to finish
- ✓ We’ll do lot’s of Django deep-dives, asking: *how does this actually work?*
- ✓ Get friendly with Django error messages
- ✓ Hands-on debugging approaches during Django development
- ✓ Understand Django projects and file structure
- ✓ Understand how requests flow through Django apps

What you’ll learn in this course

- ✓ Models, views, and templates

What you’ll learn in this course

- ✓ Models, views, and templates
- ✓ Primer on Relational Databases and using the Django ORM

What you’ll learn in this course

- ✓ Models, views, and templates
- ✓ Primer on Relational Databases and using the Django ORM
- ✓ Interact with your DB from the Django Shell and the Django Admin Interface

What you’ll learn in this course

- ✓ Models, views, and templates
- ✓ Primer on Relational Databases and using the Django ORM
- ✓ Interact with your DB from the Django Shell and the Django Admin Interface
- ✓ Using the Django Templating language and Template Inheritance

What you’ll learn in this course

- ✓ Models, views, and templates
- ✓ Primer on Relational Databases and using the Django ORM
- ✓ Interact with your DB from the Django Shell and the Django Admin Interface
- ✓ Using the Django Templating language and Template Inheritance
- ✓ URL resolving and correctly using path converters and namespaces

What you’ll learn in this course

- ✓ Models, views, and templates
- ✓ Primer on Relational Databases and using the Django ORM
- ✓ Interact with your DB from the Django Shell and the Django Admin Interface
- ✓ Using the Django Templating language and Template Inheritance
- ✓ URL resolving and correctly using path converters and namespaces
- ✓ Using Bootstrap for style and mobile responsiveness

Let’s get started!

Relax and enjoy the ride!

PART 1 - DJANGO OVERVIEW

1. About this section
2. About Web Frameworks and Django
3. What you’ll build and learn
4. **Apps in a Django project**
5. Files in a Django project
6. Flow in a Django project
7. Recap and Outlook

Apps in a Django project

PART 1 - DJANGO OVERVIEW

1. About this section
2. About Web Frameworks and Django
3. What you'll build and learn
4. Apps in a Django project
5. **Files in a Django project**
6. Flow in a Django project
7. Recap and Outlook

Files in a Django project

PART 1 - DJANGO OVERVIEW

1. About this section
2. About Web Frameworks and Django
3. What you’ll build and learn
4. Apps in a Django project
5. Files in a Django project
6. **Flow in a Django project**
7. Recap and Outlook

Flow

↓

In

↓

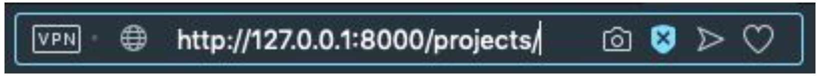
A

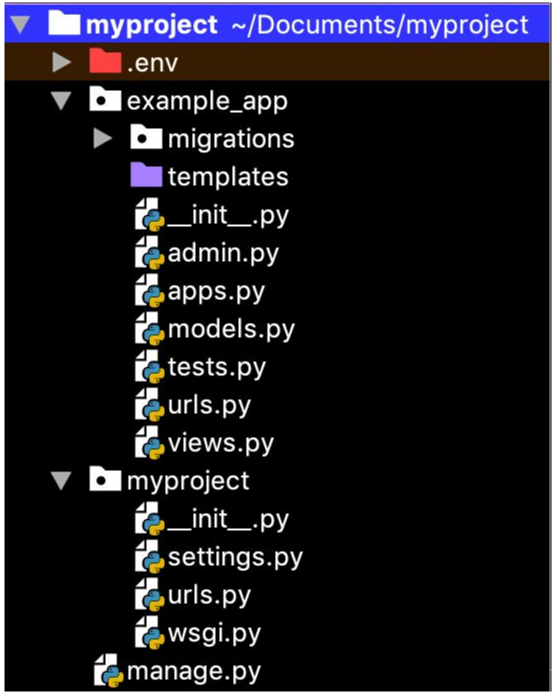
↓

Django

↓

project





```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

Flow

↓

In

↓

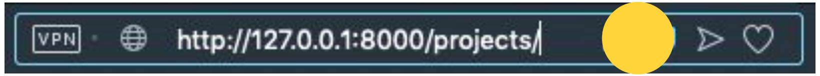
A

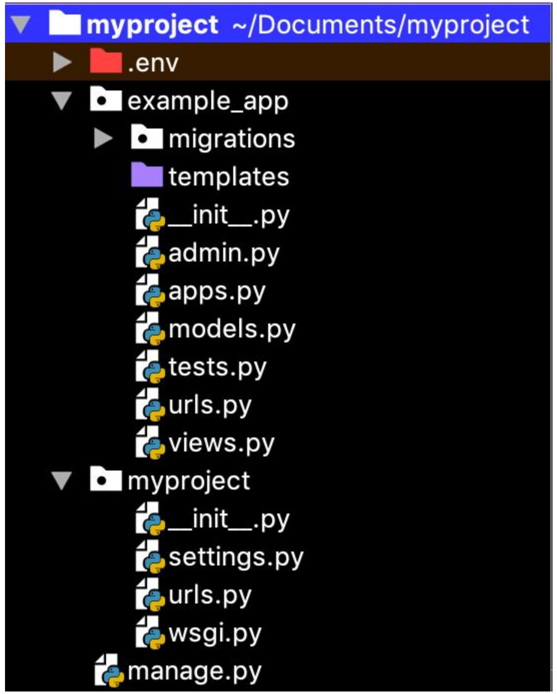
↓

Django

↓

project


http://127.0.0.1:8000/projects/


```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

Flow

↓

In

↓

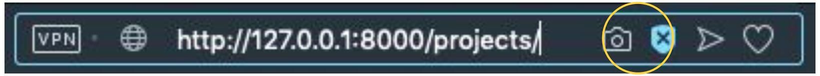
A

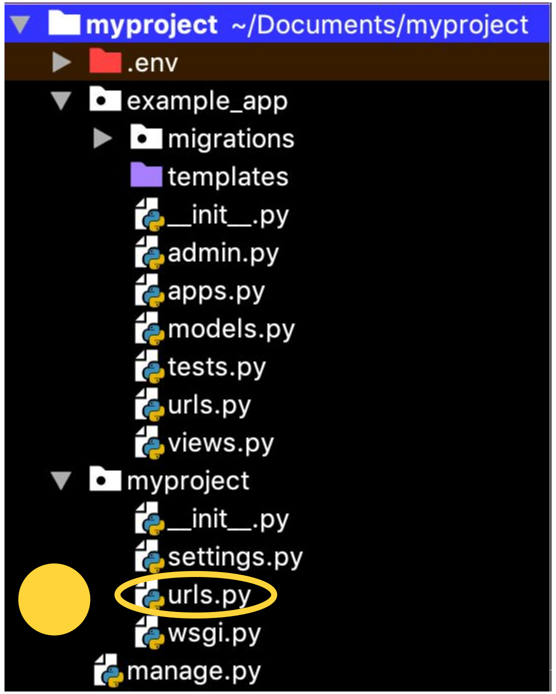
↓

Django

↓

project


VPN http://127.0.0.1:8000/projects/


```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

Flow

↓

In

↓

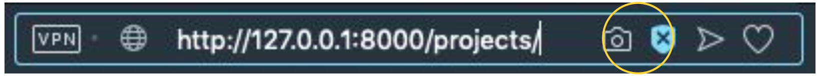
A

↓

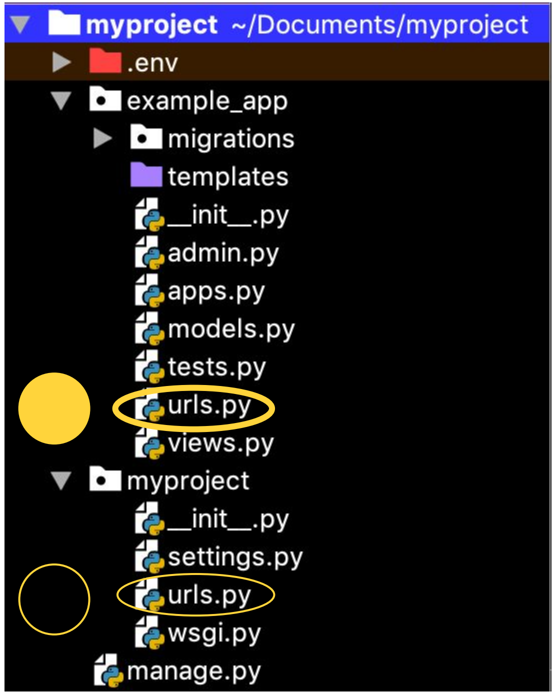
Django

↓

project

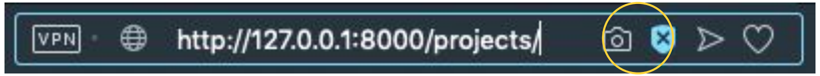


http://127.0.0.1:8000/projects/

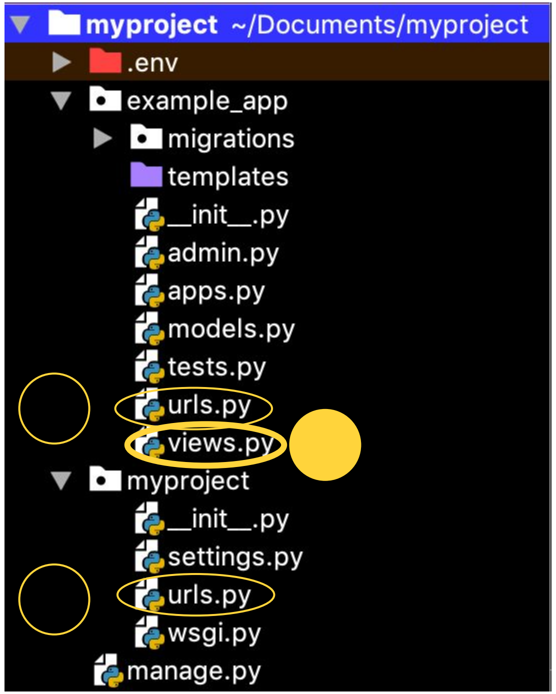


```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

Flow
↓
In
↓
A
↓
Django
↓
project



http://127.0.0.1:8000/projects/



```
myproject ~/Documents/myproject
  .env
  example_app
    migrations
    templates
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  myproject
    __init__.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```

Flow

↓

In

↓

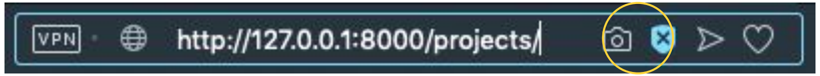
A

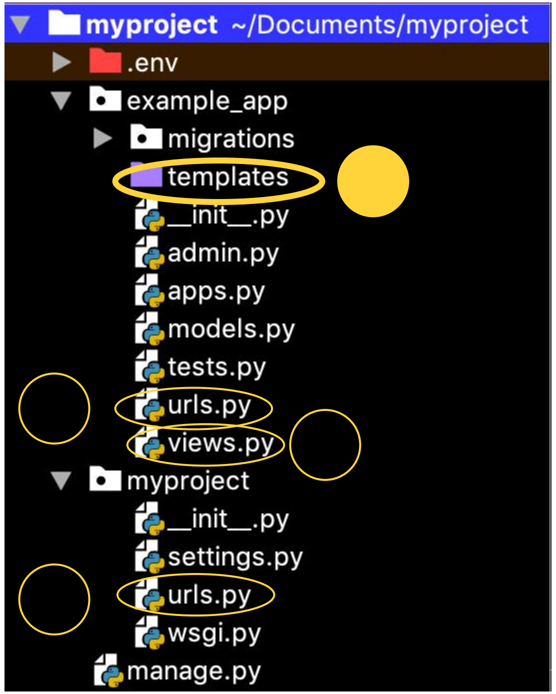
↓

Django

↓

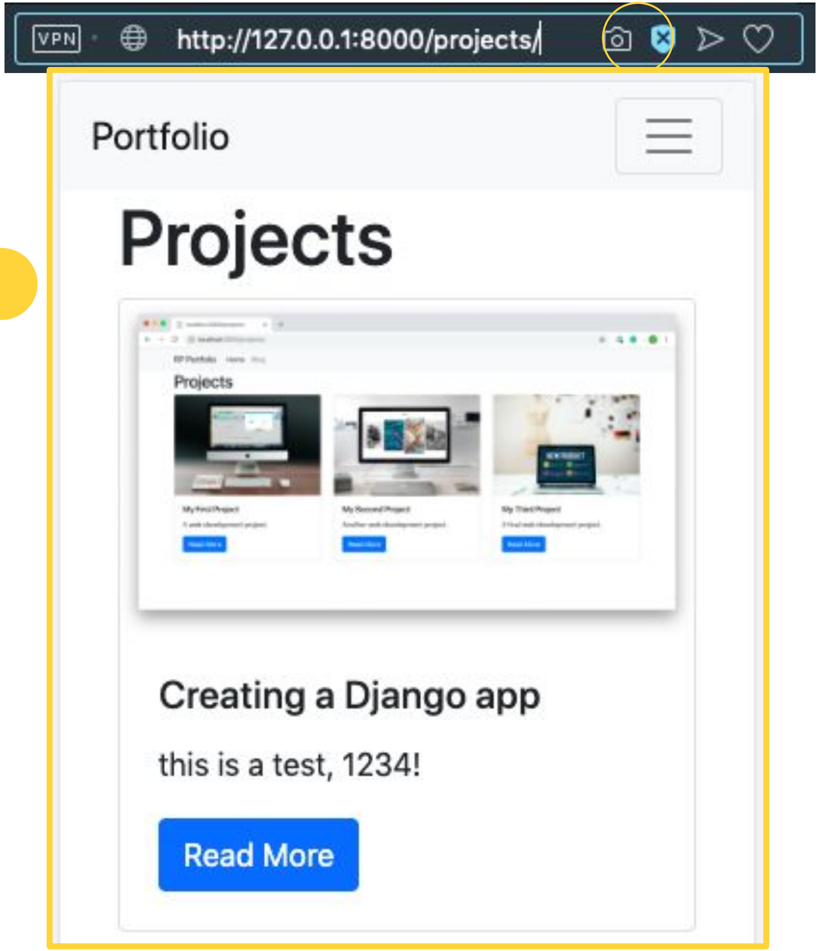
project





```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

Flow ↓ In ↓ A ↓ Django ↓ project


http://127.0.0.1:8000/projects/

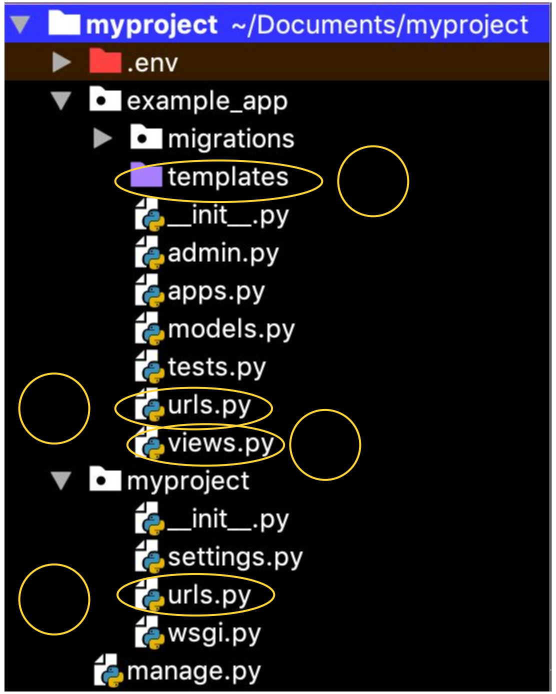
Portfolio

Projects

Creating a Django app

this is a test, 1234!

Read More


```
myproject ~/Documents/myproject
  .env
  example_app
    migrations
    templates
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  myproject
    __init__.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```

Flow

↓

In

↓

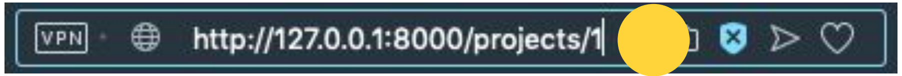
A

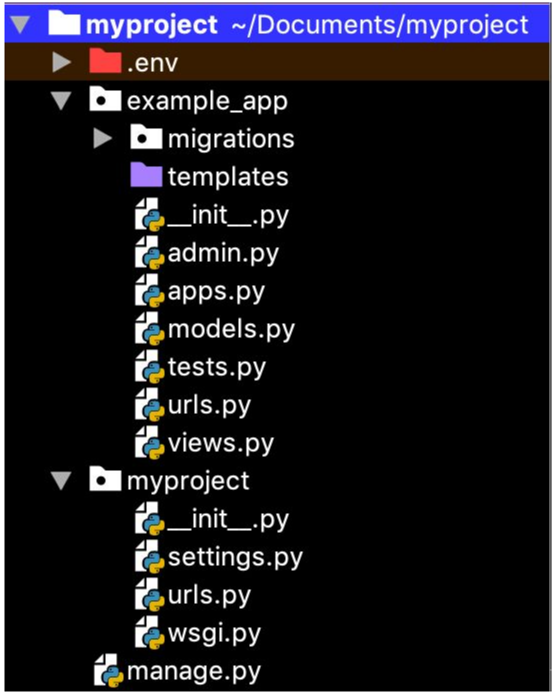
↓

Django

↓

project


http://127.0.0.1:8000/projects/1


```
▼ myproject ~/Documents/myproject
    ▶ .env
    ▼ example_app
        ▶ migrations
          templates
          __init__.py
          admin.py
          apps.py
          models.py
          tests.py
          urls.py
          views.py
    ▼ myproject
          __init__.py
          settings.py
          urls.py
          wsgi.py
      manage.py
```

Flow

↓

In

↓

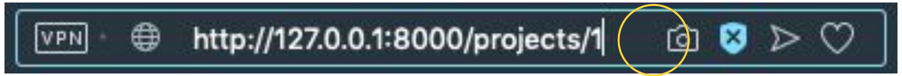
A

↓

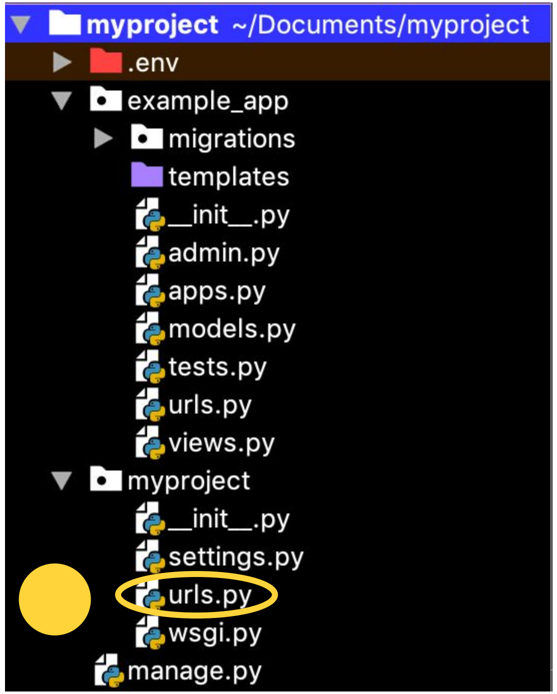
Django

↓

project



http://127.0.0.1:8000/projects/1



```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

Flow

↓

In

↓

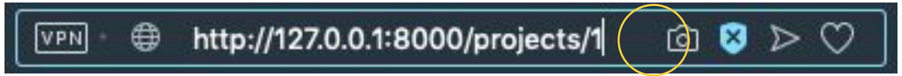
A

↓

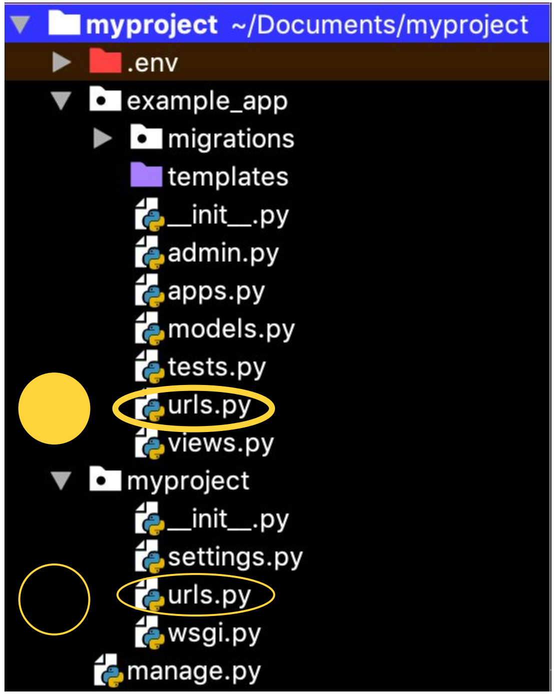
Django

↓

project



http://127.0.0.1:8000/projects/1



```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

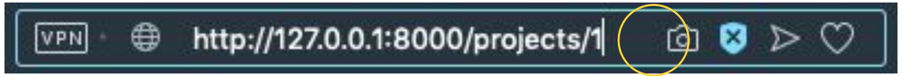
Flow

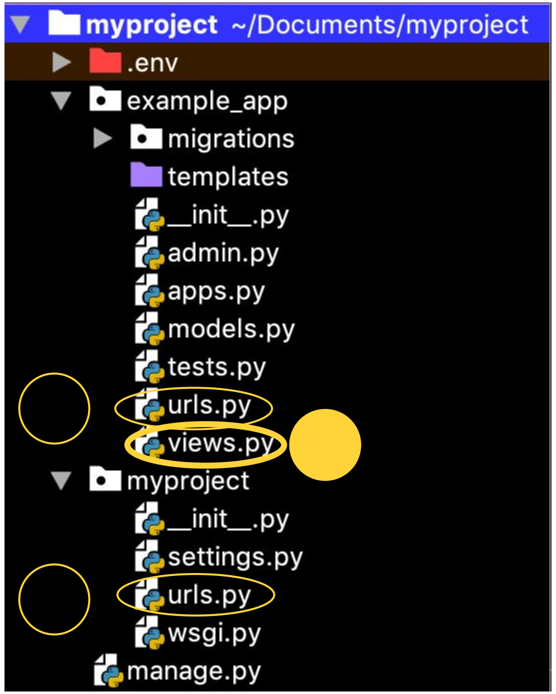
In

A

Django

project


http://127.0.0.1:8000/projects/1


```
myproject ~/Documents/myproject
  .env
  example_app
    migrations
    templates
    __init__.py
    admin.py
    apps.py
    models.py
    tests.py
    urls.py
    views.py
  myproject
    __init__.py
    settings.py
    urls.py
    wsgi.py
  manage.py
```

Flow

↓

In

↓

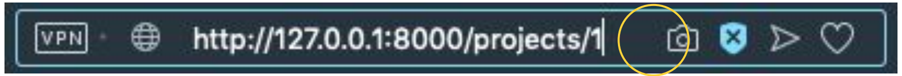
A

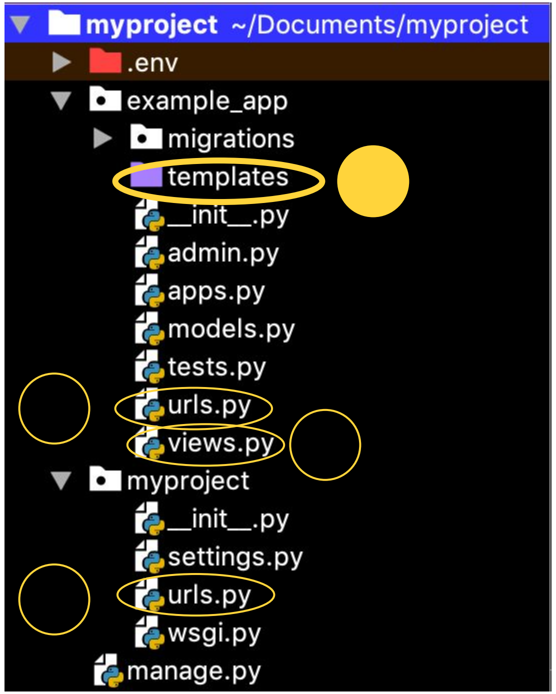
↓

Django

↓

project


http://127.0.0.1:8000/projects/1


```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

Flow

↓

In

↓

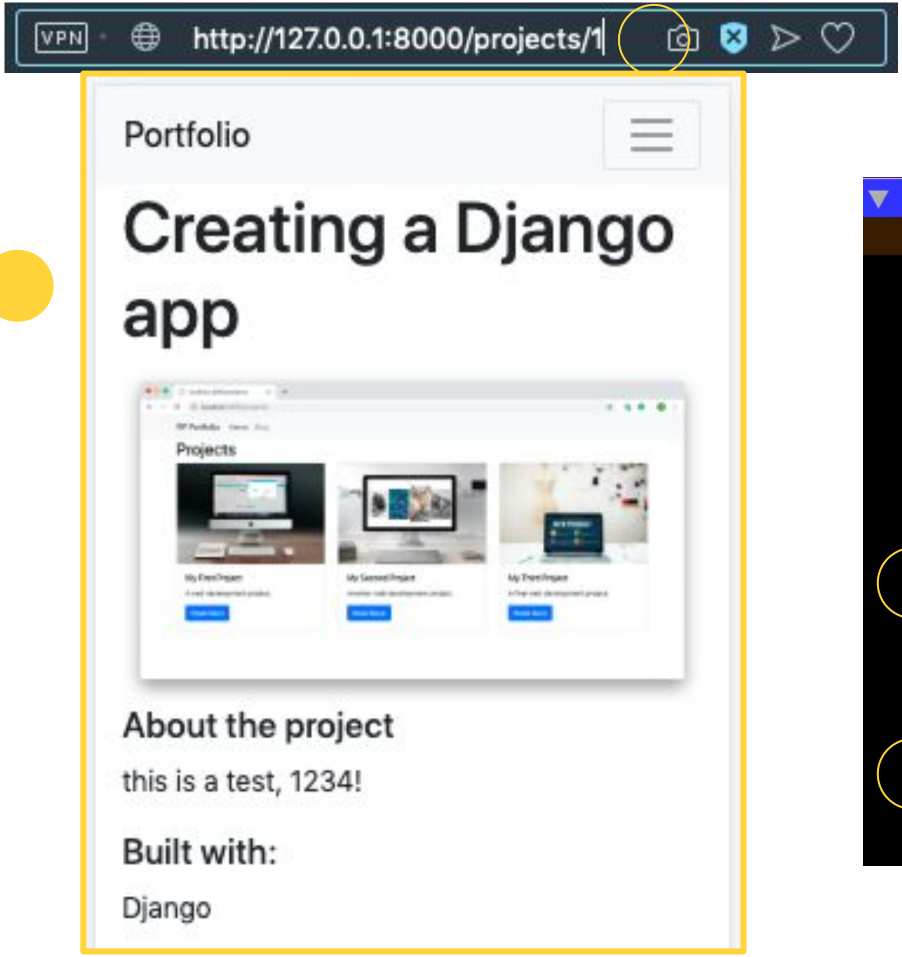
A

↓

Django

↓

project



http://127.0.0.1:8000/projects/1

Portfolio

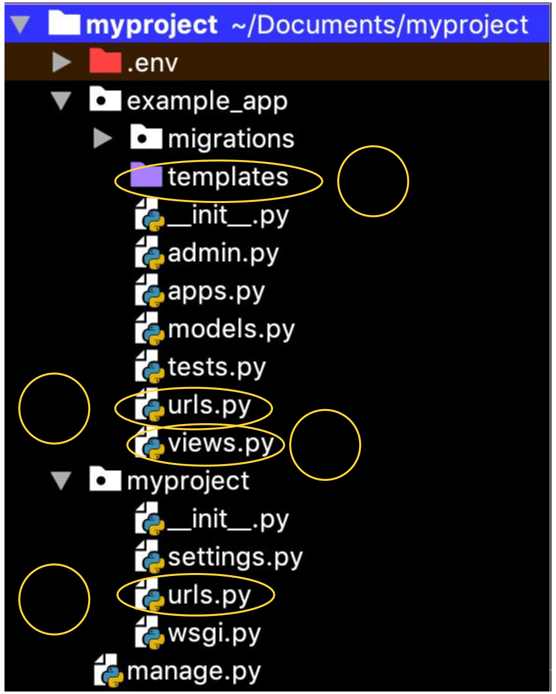
Creating a Django app

About the project

this is a test, 1234!

Built with:

Django



```
▼ myproject ~/Documents/myproject
  ▶ .env
  ▼ example_app
    ▶ migrations
      templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  ▼ myproject
      __init__.py
      settings.py
      urls.py
      wsgi.py
    manage.py
```

PART 1 - DJANGO OVERVIEW

1. About this section
2. About Web Frameworks and Django
3. What you’ll build and learn
4. Apps in a Django project
5. Files in a Django project
6. Flow in a Django project
7. **Recap and Outlook**

What you did in this section

✓ Got an overview of the upcoming video course content

What you did in this section

- ✓ Got an overview of the upcoming video course content
- ✓ Saw the file structure we’ll be learning to understand

What you did in this section

- ✓ Got an overview of the upcoming video course content
- ✓ Saw the file structure we'll be learning to understand
- ✓ Heard about the flow of a request through a Django app

What you did in this section

- ✓ Got an overview of the upcoming video course content
- ✓ Saw the file structure we'll be learning to understand
- ✓ Heard about the flow of a request through a Django app
- ✓ Understood the pluggable nature of Django apps in projects

What you did in this section

- ✓ Got an overview of the upcoming video course content
- ✓ Saw the file structure we'll be learning to understand
- ✓ Heard about the flow of a request through a Django app
- ✓ Understood the pluggable nature of Django apps in projects

PART 2 - SET UP YOUR DEVELOPMENT ENVIRONMENT

What you need to continue

- ✓ Python 3.x installed
- ✓ A text editor or an IDE (e.g. PyCharm)
- ✓ Beginner knowledge of using the CLI (Terminal)

PART 2 - SET UP YOUR DEV. ENVIRONMENT

▶ 1. **About this section**

2. Set up your development environment
    a. Using PyCharm
    b. Creating a Virtual Environment
    c. Installing Django
3. Create a Django project
4. Recap and Outlook

Create a Virtual Environment

Install Django using pip

PART 2 - SET UP YOUR DEV. ENVIRONMENT

1. About this section
2. **Set up your development environment**
   a. Using PyCharm
   b. Creating a Virtual Environment
   c. Installing Django
3. Create a Django project
4. Recap and Outlook

PART 2 - SET UP YOUR DEV. ENVIRONMENT

1. About this section
2. Set up your development environment
    a. Using PyCharm
    b. Creating a Virtual Environment
    c. Installing Django
3. **Create a Django project**
4. Recap and Outlook

PART 2 - SET UP YOUR DEV. ENVIRONMENT

1. About this section
2. Set up your development environment
   a. Using PyCharm
   b. Creating a Virtual Environment
   c. Installing Django
3. Create a Django project
4. **Recap and Outlook**

What you did in this section

✓ Created a Virtual Environment in the CLI

```
python3 -m venv .env
```

✓ Activated the venv

```
source .env/bin/activate
```

What you did in this section

✓ Installed Django inside the venv

```
(.env)$ pip install django
```

What you did in this section

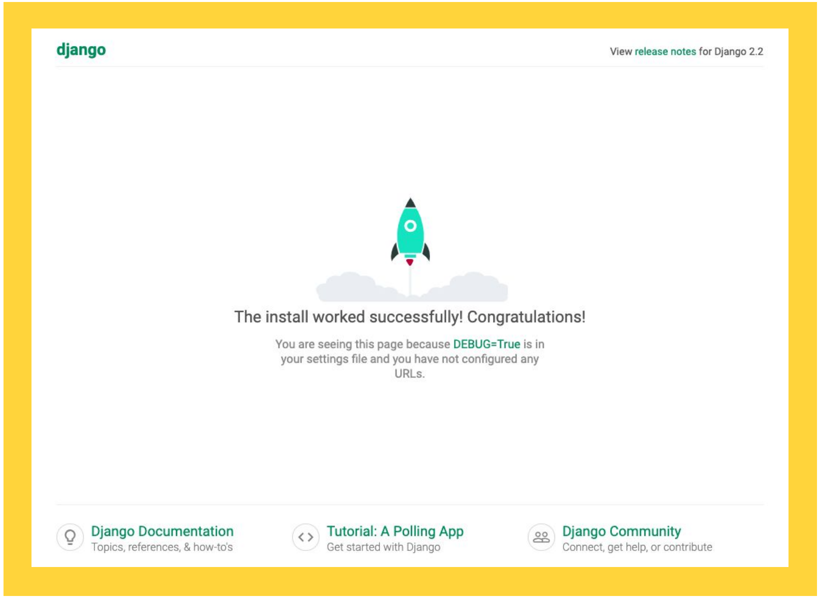
✓ Created a Django project management app inside the existing folder

```
(.env)$ django-admin startproject portfolio .
```

The dot avoids extra folders!

What you did in this section

✓ Visited the site at **http://localhost:8000**



django

View release notes for Django 2.2

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation
Topics, references, & how-to's

Tutorial: A Polling App
Get started with Django

Django Community
Connect, get help, or contribute

PART 3 - BUILD A DJANGO APPLICATION

What you need to continue

- ✓ Set up: Python 3.x, Virtual Environment, Django, text editor
- ✓ Created a Django project called “portfolio”
- ✓ Motivation to build a webapp :)

PART 3 - BUILD A DJANGO APPLICATION

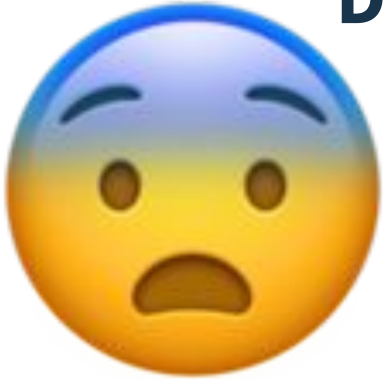
▶ 1. **About this section**
2. Django by error messages
3. Create a Django app
4. Build your routes
5. Create a view
6. Create a template
7. Add bootstrap to your app
8. Recap and Outlook

PART 3 - BUILD A DJANGO APPLICATION

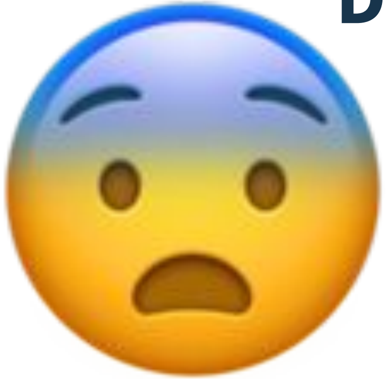
1. About this section
2. **Django by error messages**
3. Create a Django app
4. Build your routes
5. Create a view
6. Create a template
7. Add bootstrap to your app
8. Recap and Outlook

Developing by error messages

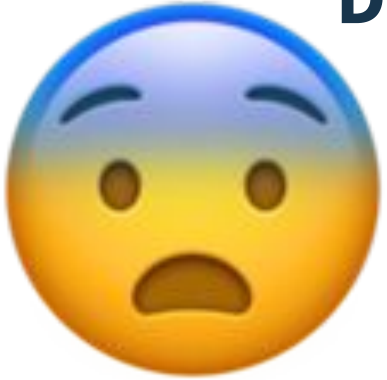
Developing by error messages



Developing by error messages



Developing by error messages



Developing by error messages

Error messages are your friends!

Developing by error messages

Developing by error messages

Developing by error messages

A Django error message

Error messages are your friends!

PART 3 - BUILD A DJANGO APPLICATION

1. About this section
2. Django by error messages
3. **Create a Django app**
4. Build your routes
5. Create a view
6. Create a template
7. Add bootstrap to your app
8. Recap and Outlook

Django settings

Time to meet a new friend!

PART 3 - BUILD A DJANGO APPLICATION

1. About this section
2. Django by error messages
3. Create a Django app
4. **Build your routes**
5. Create a view
6. Create a template
7. Add bootstrap to your app
8. Recap and Outlook

Time to meet a new friend!

PART 3 - BUILD A DJANGO APPLICATION

1. About this section
2. Django by error messages
3. Create a Django app
4. Build your routes
5. **Create a view**
6. Create a template
7. Add bootstrap to your app
8. Recap and Outlook

Time to meet a new friend!

Sneaking...

PART 3 - BUILD A DJANGO APPLICATION

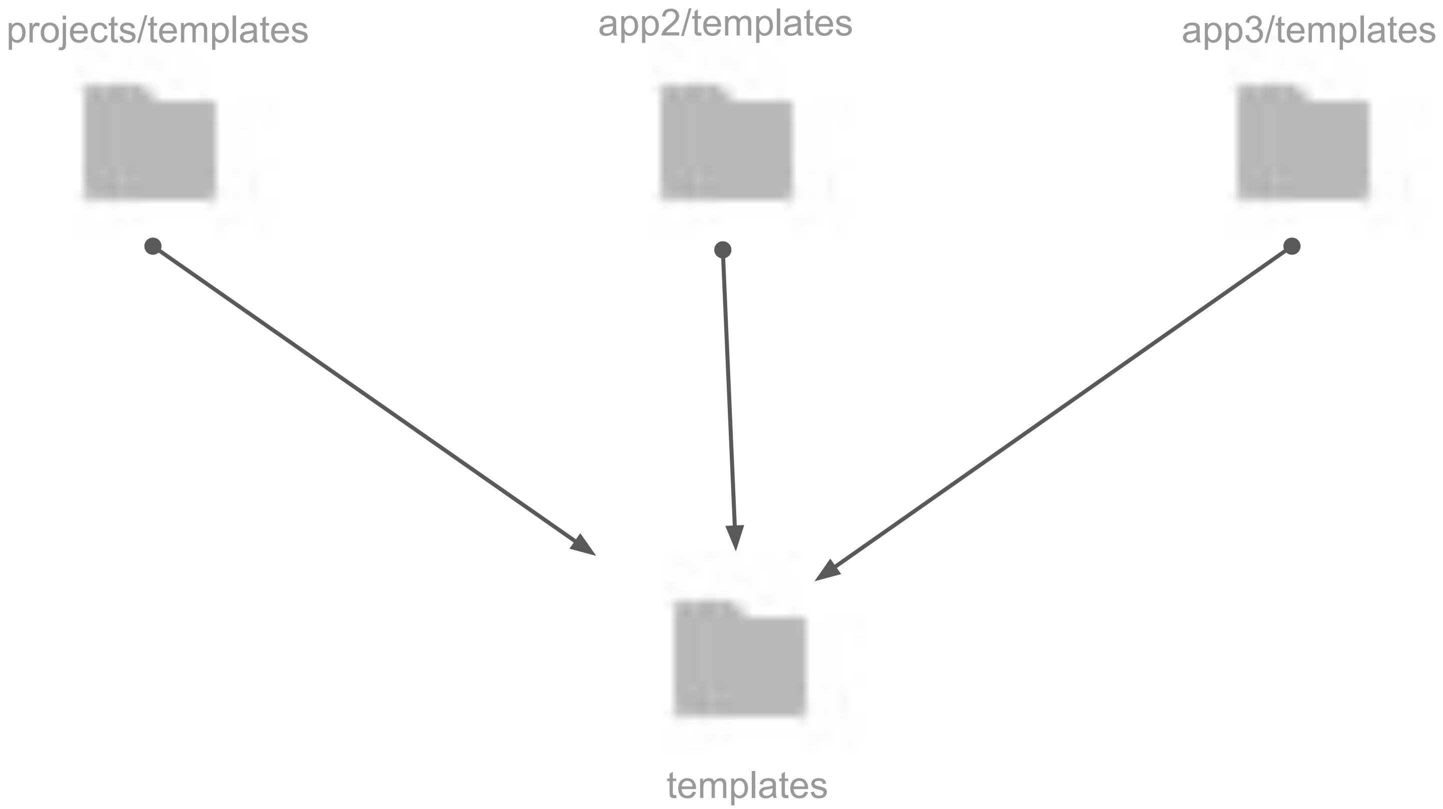
1. About this section
2. Django by error messages
3. Create a Django app
4. Build your routes
5. Create a view
6. **Create a template**
7. Add bootstrap to your app
8. Recap and Outlook

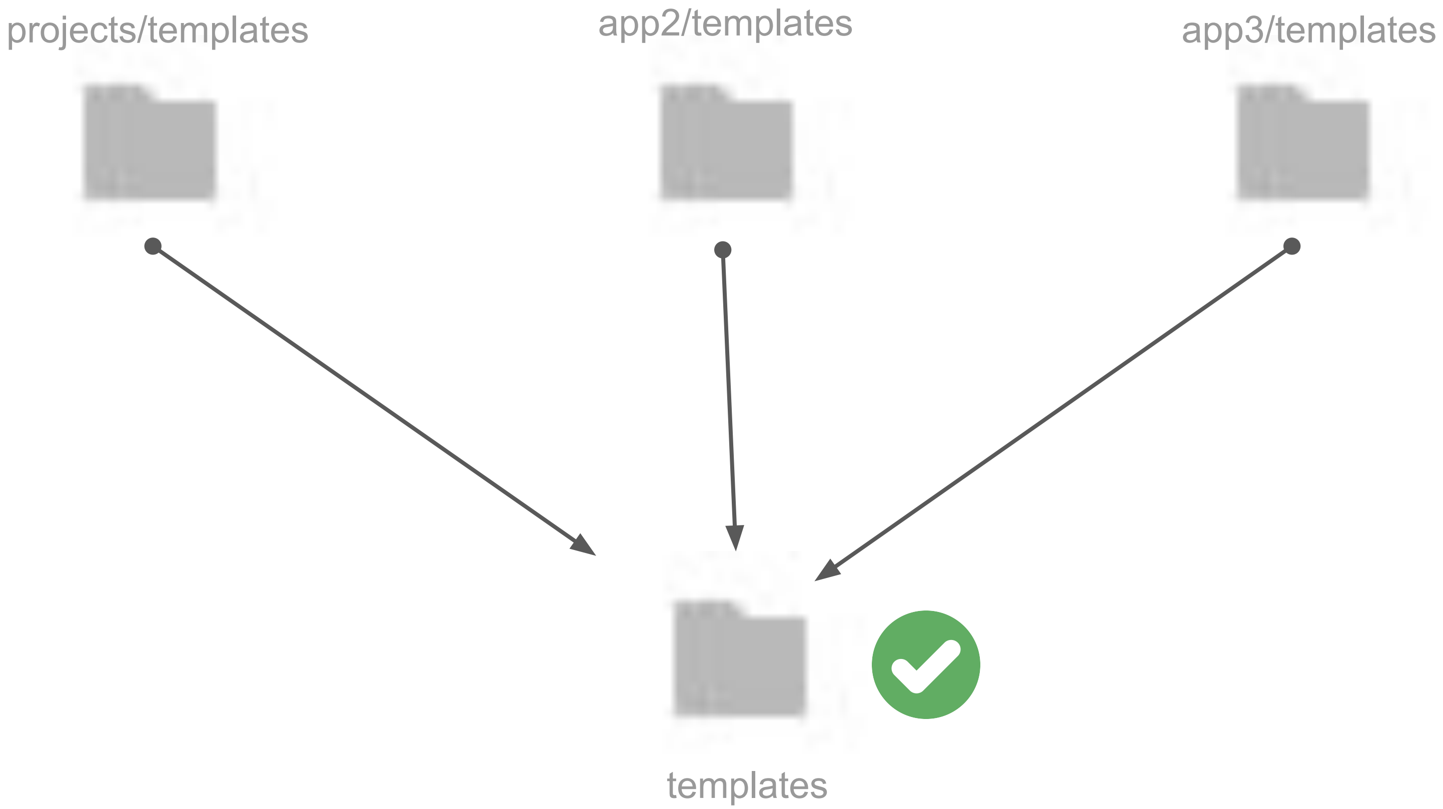
Flat folder structure

projects/templates

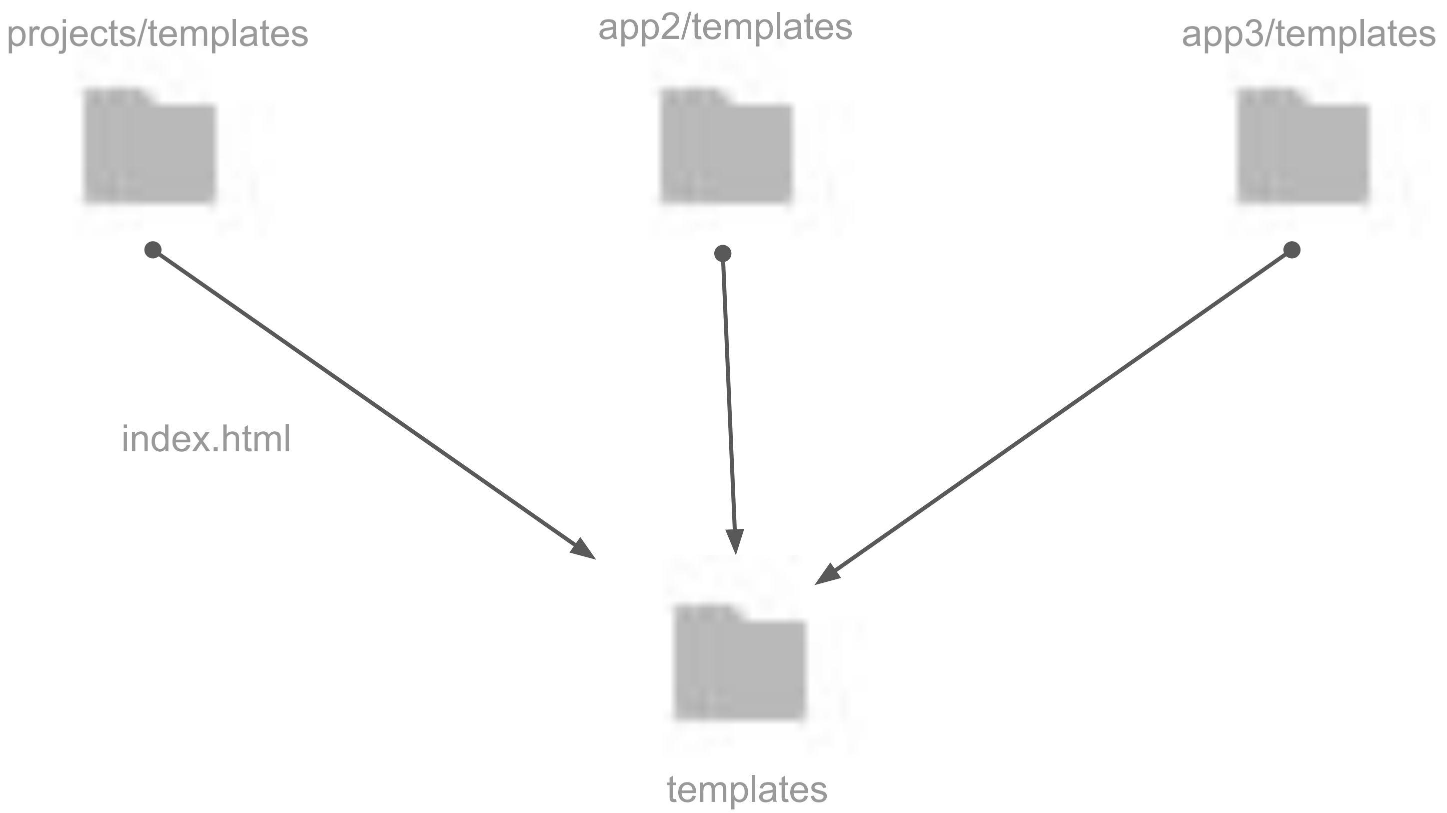
app2/templates

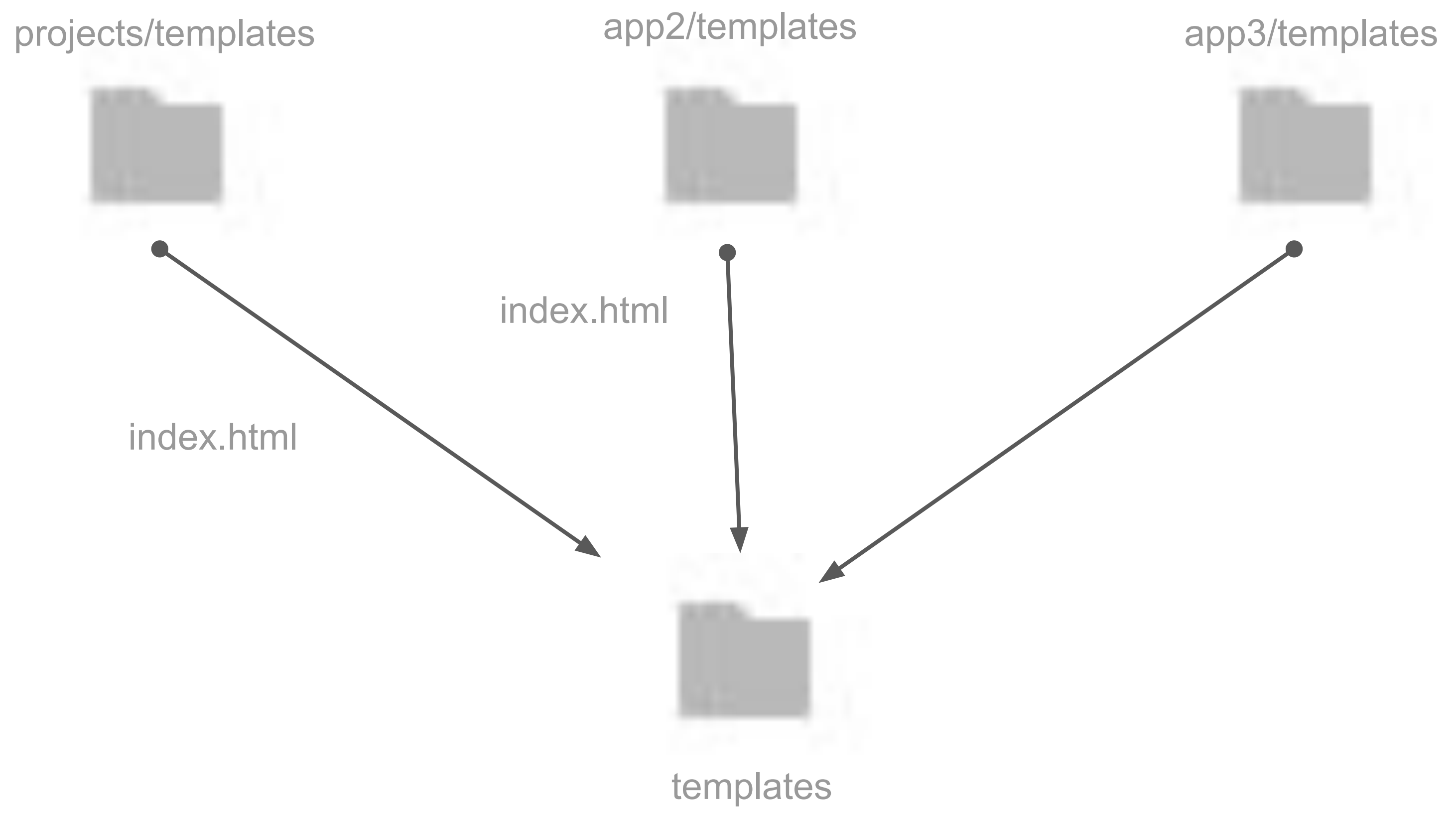
app3/templates

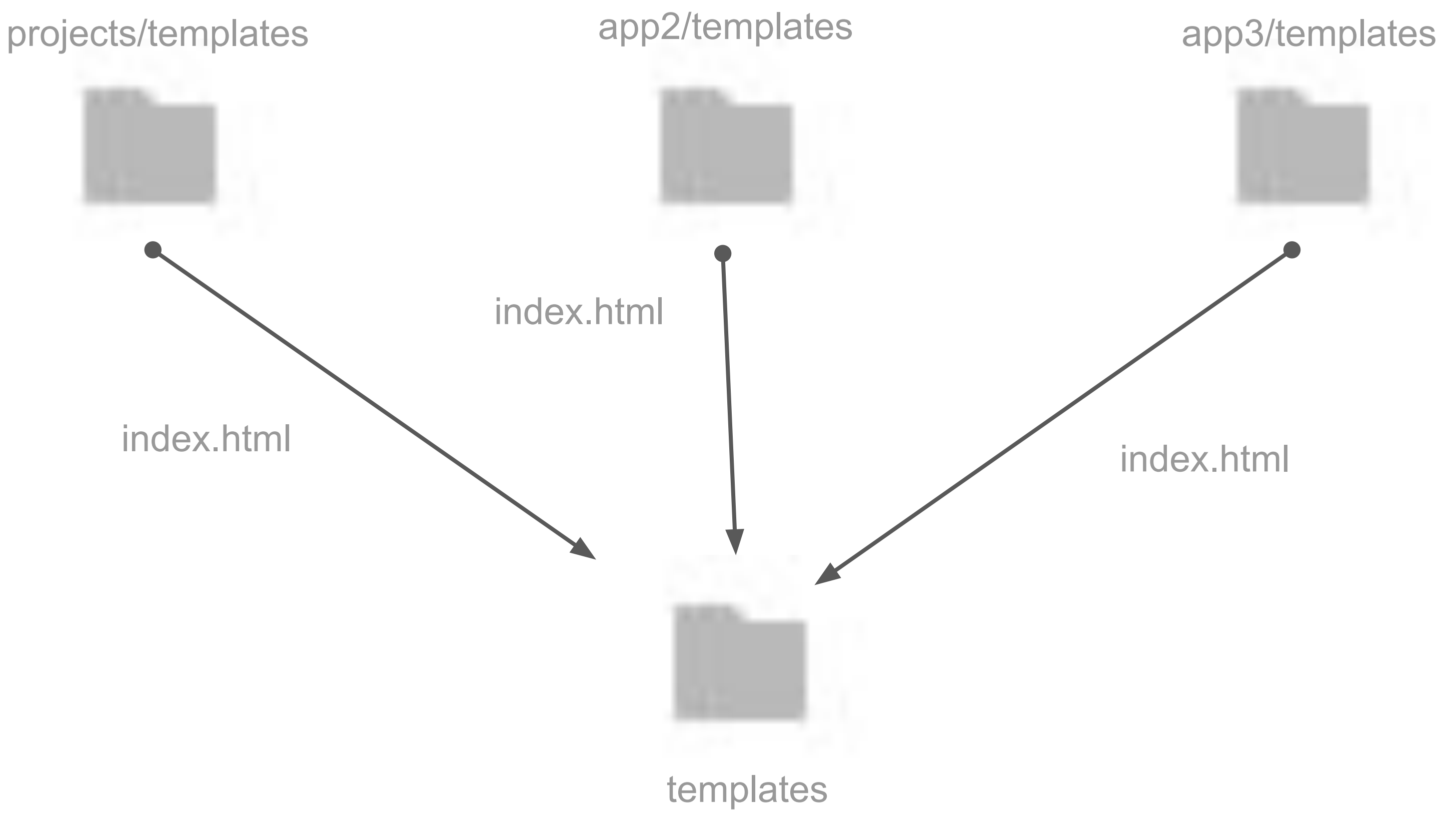

projects/templates
app2/templates
app3/templates
templates

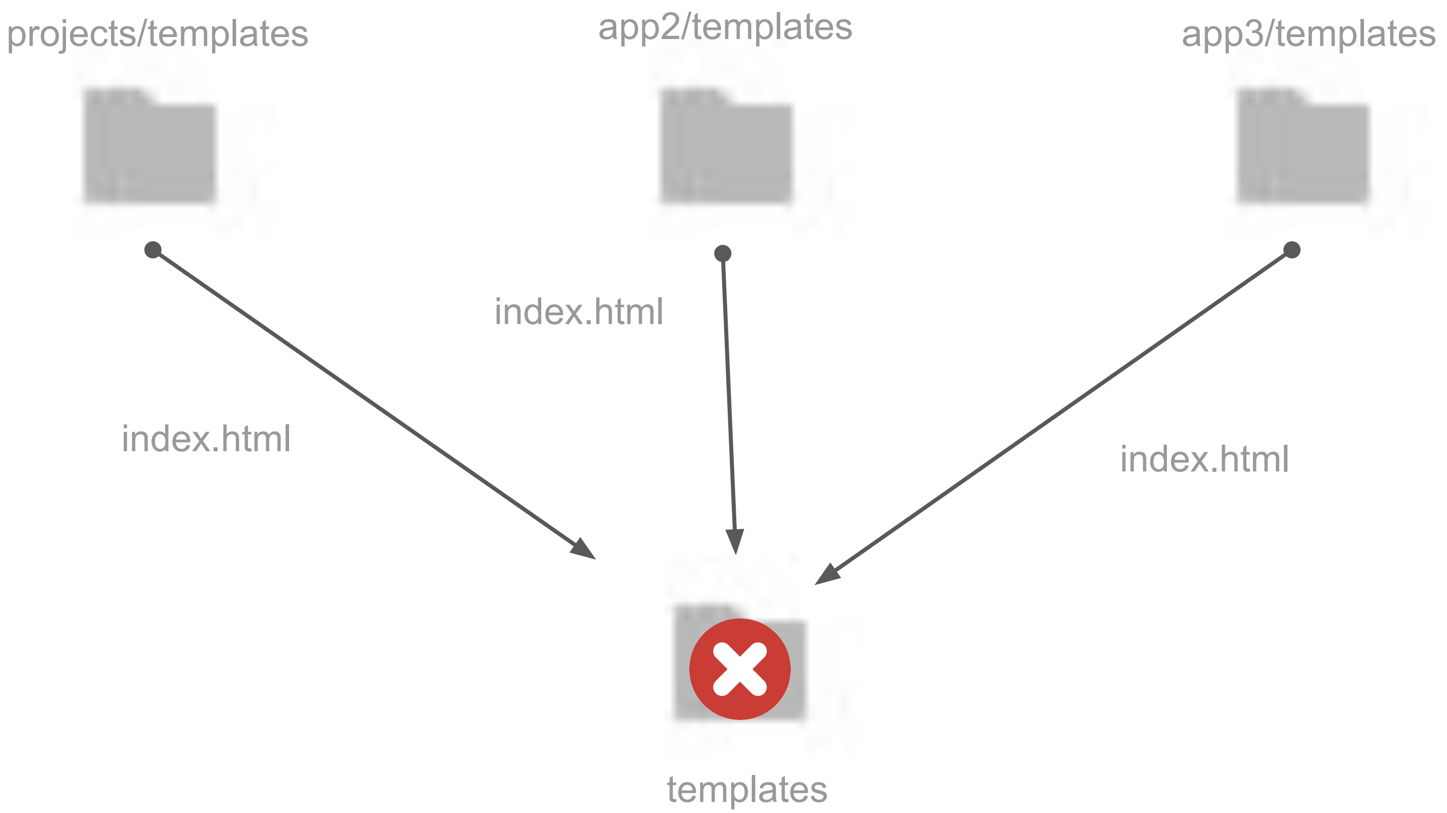

projects/templates
app2/templates
app3/templates
templates

Troubles with flat folder structure

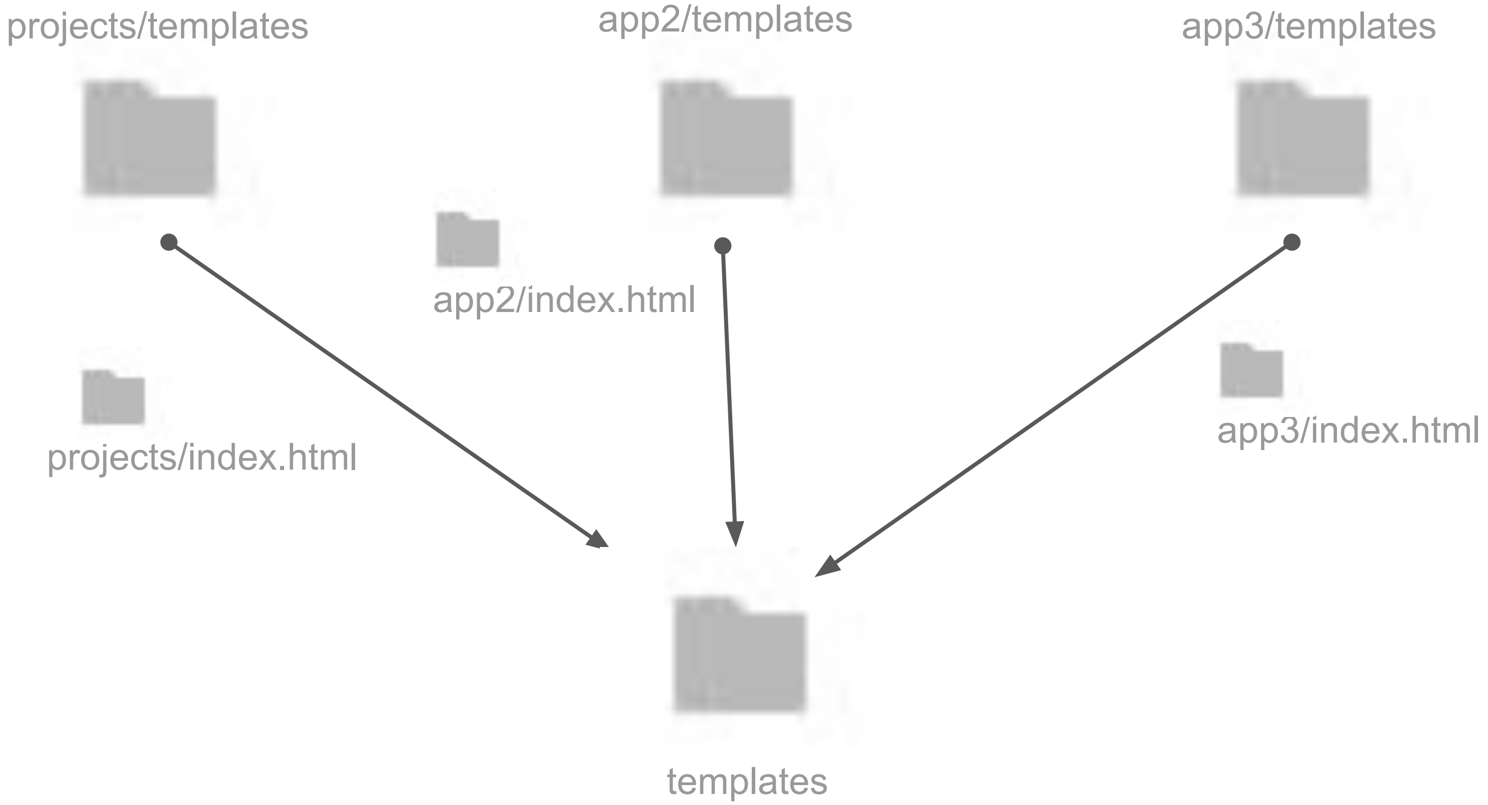

projects/templates
app2/templates
app3/templates
index.html
templates

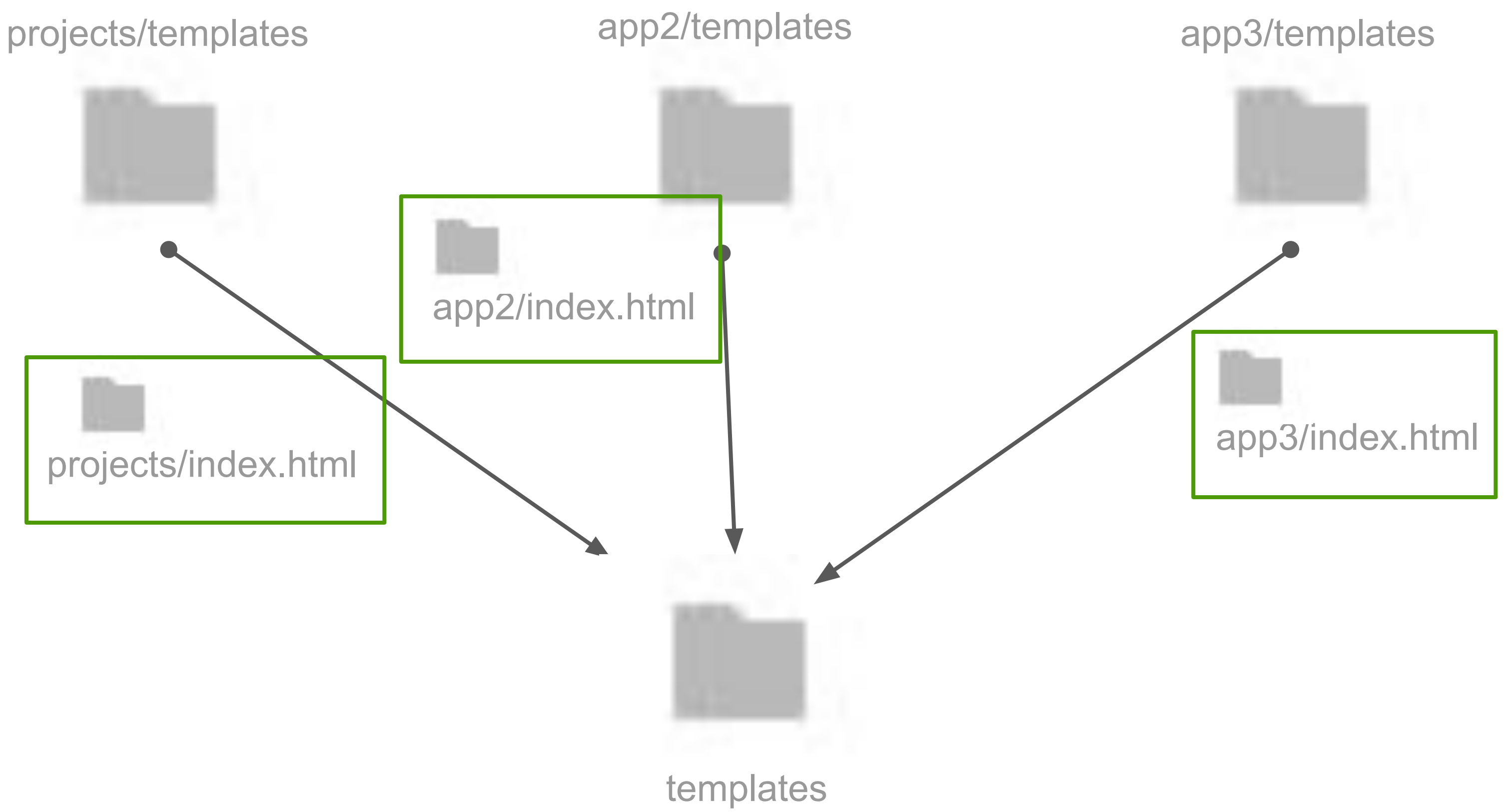

projects/templates
app2/templates
app3/templates
index.html
index.html
templates

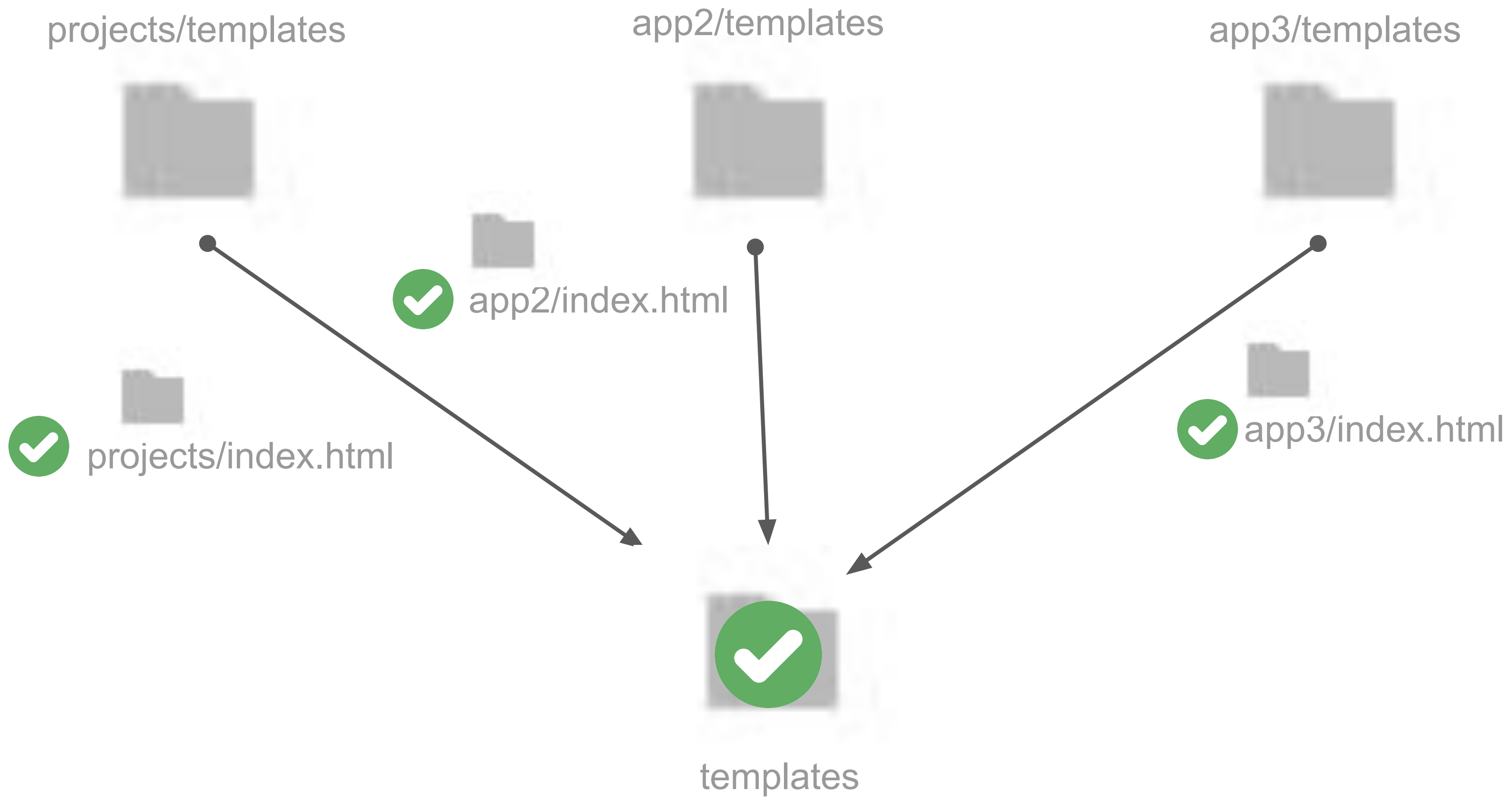

projects/templates
app2/templates
app3/templates
index.html
index.html
index.html
templates

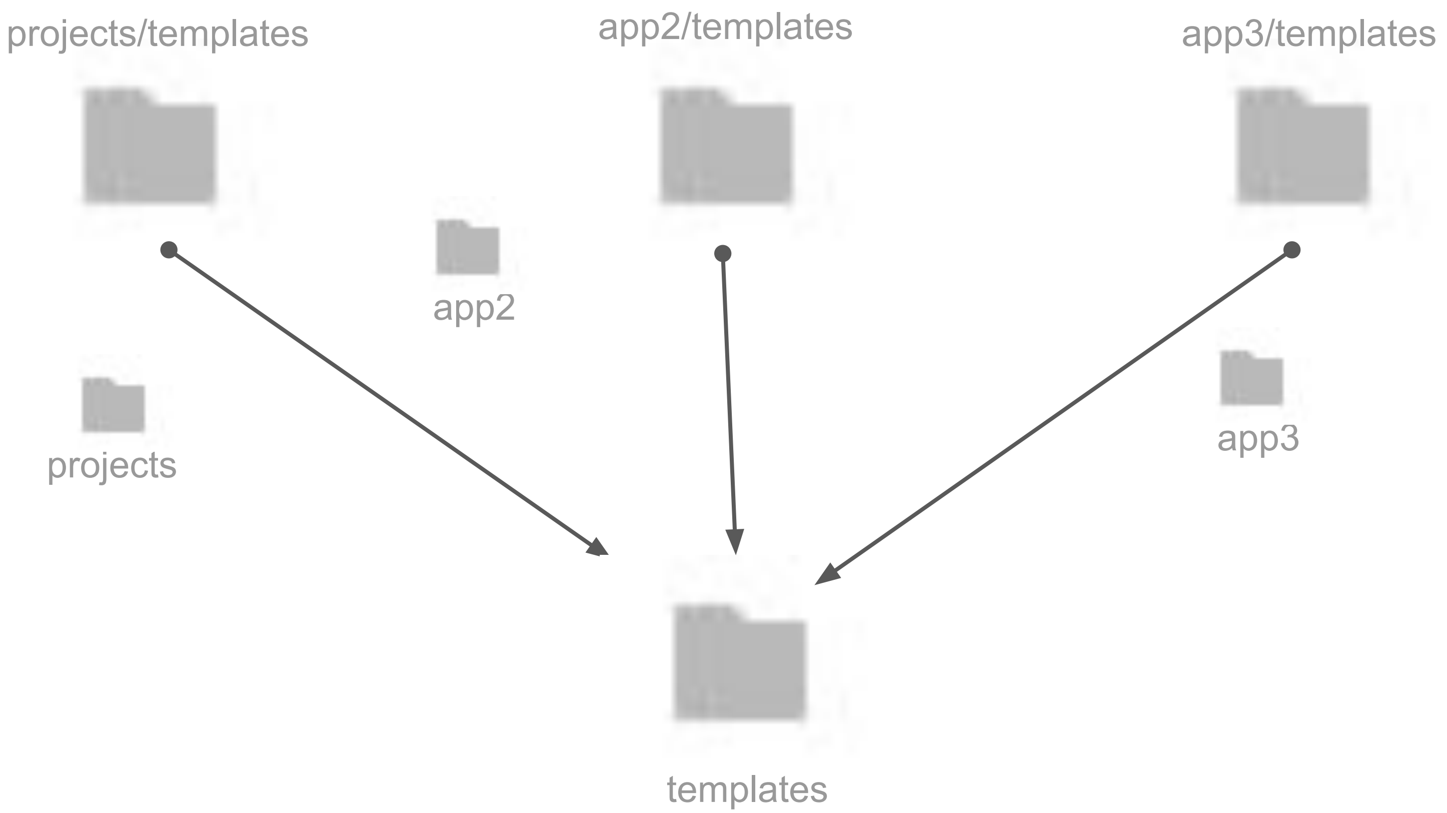

projects/templates
app2/templates
app3/templates
index.html
index.html
index.html
templates

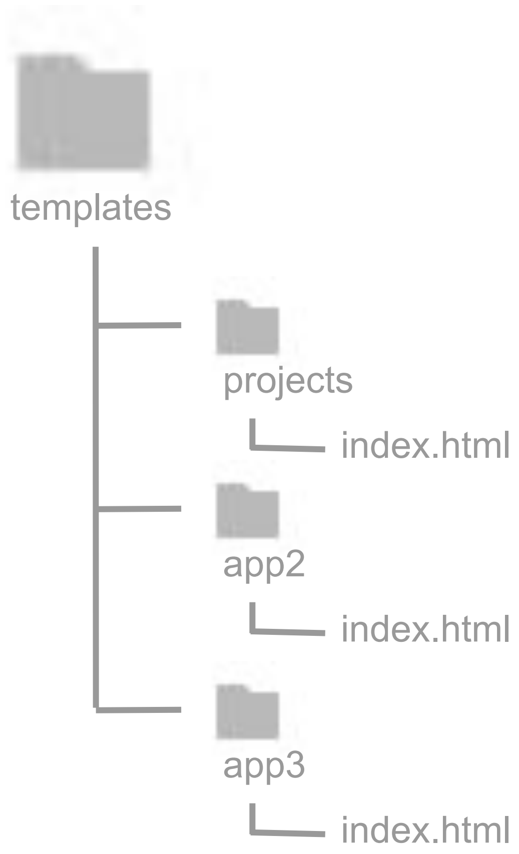
Django’s double-folder structure


projects/templates
app2/templates
app3/templates
app2/index.html
projects/index.html
app3/index.html
templates


projects/templates
app2/templates
app3/templates
app2/index.html
projects/index.html
app3/index.html
templates


projects/templates
app2/templates
app3/templates
app2/index.html
projects/index.html
app3/index.html
templates


projects/templates
app2/templates
app3/templates
app2
projects
app3
templates


templates
projects
index.html
app2
index.html
app3
index.html

Time to meet a new friend!

PART 3 - BUILD A DJANGO APPLICATION

1. About this section
2. Django by error messages
3. Create a Django app
4. Build your routes
5. Create a view
6. Create a template
7. **Add bootstrap to your app**
8. Recap and Outlook

PART 3 - BUILD A DJANGO APPLICATION

1. About this section
2. Django by error messages
3. Create a Django app
4. Build your routes
5. Create a view
6. Create a template
7. Add bootstrap to your app
8. **Recap and Outlook**

What you did in this section

✓ Created a Django app called “projects”

```
(.env)$ python manage.py startapp projects
```

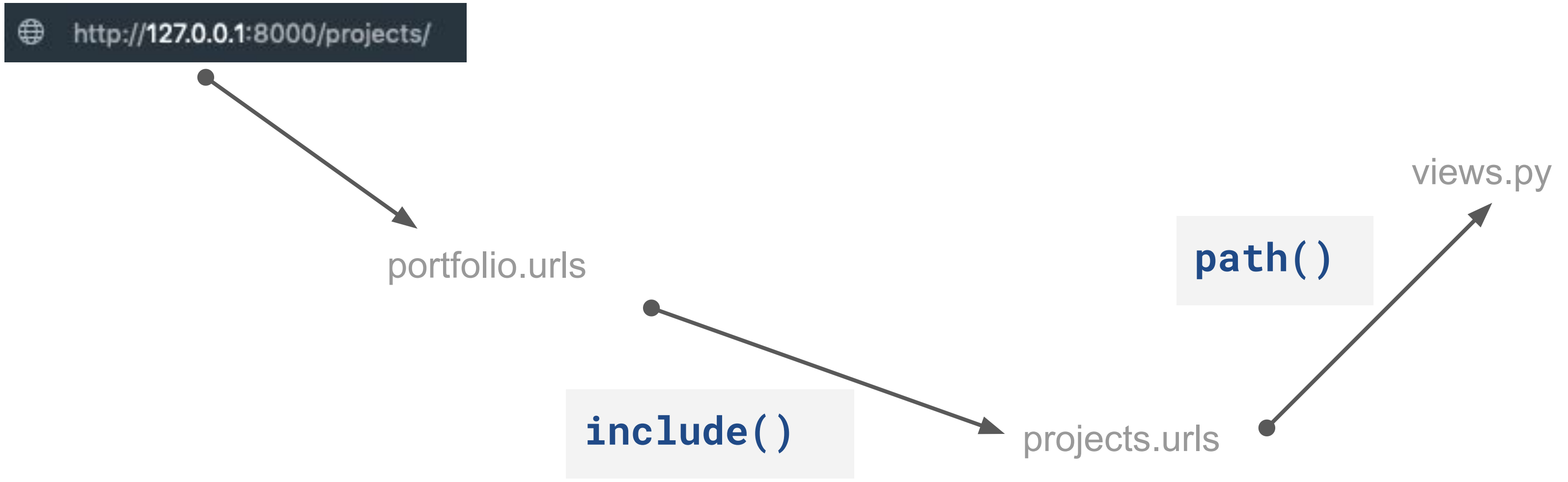
What you did in this section

✓ Registered your app in settings.py

```
INSTALLED_APPS = [
    # django apps
    # my apps
    "projects",
]
```

What you did in this section

✓ Dug your way through different URL configurations



What you did in this section

✓ Created a view function that returns a HttpResponse object

```python
def project_list(request):
    return HttpResponse("<h1>Aye!</h1>")
```

What you did in this section

✓ Changed the view function to render a template instead

```python
def project_list(request):
    return render(request, "projects/index.html")
```

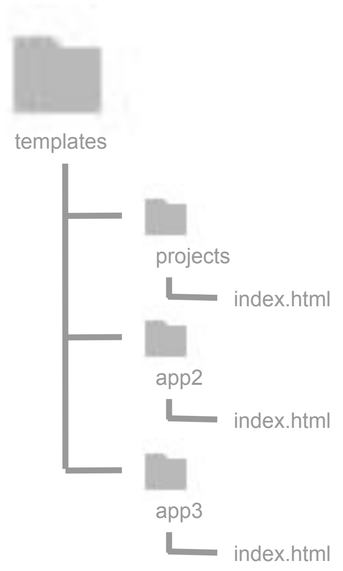
What you did in this section

✓ Created a Django template containing HTML code

```
<body>
    <h1>Hello template!</h1>
</body>
```

What you did in this section

✓ Understood the nested template folder structure in Django



What you did in this section

✓ Registered your new template folder in settings.py

```
TEMPLATES = [
    {
        "DIRS": [os.path.join(BASE_DIR, "projects/templates"),]
    }
]
```

What you did in this section

✓ Added Bootstrap styling to your app

```html
<head>
    <link rel="stylesheet" href="link-to-bootstrap-CDN/bootstrap.css">
</head>
```

And… most importantly:

You made a lot of new friends!!!

- ✓ PageNotFound
- ✓ ModuleNotFoundError
- ✓ ImproperlyConfigured
- ✓ AttributeError
- ✓ ValueError
- ✓ TemplateDoesNotExist

What you did in this section

- ✓ Developing by error messages
- ✓ Getting familiar with **common Django error messages**
- ✓ Practiced to **read and follow helpful suggestions**
- ✓ Practiced to **debug** your code with the help of error messages

Error messages are your friends!

PART 4: SHOWCASE YOUR PROJECTS

What you need to continue

- ✓ Set up: Python 3.x, Virtual Environment, Django, text editor
- ✓ Django project called “portfolio”
- ✓ Django app called “projects”
- ✓ urls.py file in your “projects” app and correct routing
- ✓ double-templates folder setup

PART 4 - SHOWCASE YOUR PROJECTS

1. **About this section**
2. Django Models
3. Migrations and your SQL Database
4. Add static files
5. Use the Django Shell
6. Build your routes
7. Create your views
8. Create your templates
9. Add some style!
10. Recap and Outlook

Seriously serious!

PREVIEW: Projects app

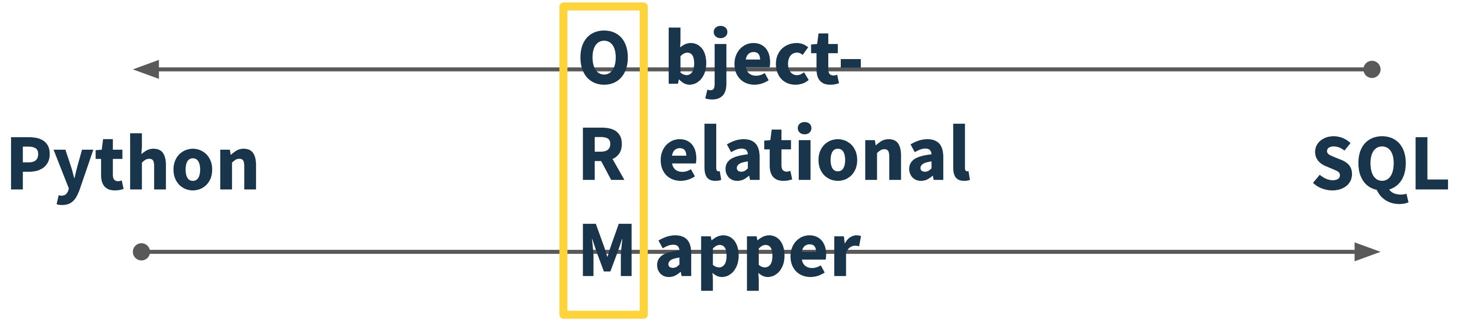
PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. **Django Models**
3. Migrations and your SQL Database
4. Add static files
5. Use the Django Shell
6. Build your routes
7. Create your views
8. Create your templates
9. Add some style!
10. Recap and Outlook

Django ORM

Object-Relational Mapper

O bject-
R elational
M apper


Object-
Relational
Mapper
Python
SQL

A (very) short intro to Relational DBs

That’s a table!

Project table

id	title	description	technology	image
1	test	This is a test	Django	/img/t.png
2	daily	Write words	Django	/img/d.png
3	todo	What to do?	Tkinter	/img/td.png

That’s a table!

Project table

id	title	description	technology	image
1	test	This is a test	Django	/img/t.png
2	daily	Write words	Django	/img/d.png
3	todo	What to do?	Tkinter	/img/td.png

Row

That’s a table!

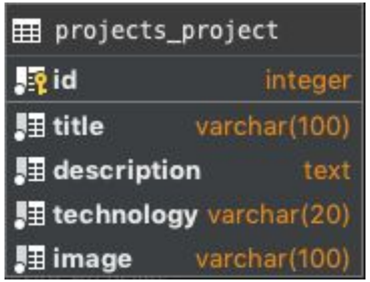
Column

Project table

id	title	description	technology	image
1	test	This is a test	Django	/img/t.png
2	daily	Write words	Django	/img/d.png
3	todo	What to do?	Tkinter	/img/td.png

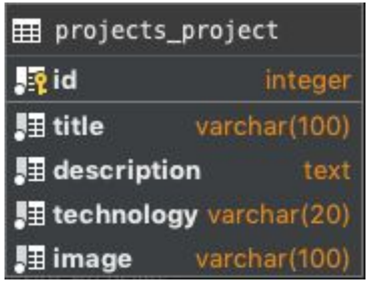
```python
class Project(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    technology = models.CharField(max_length=20)
    image = models.FilePathField(path="/img")
```

```python
class Project(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    technology = models.CharField(max_length=20)
    image = models.FilePathField(path="/img")
```

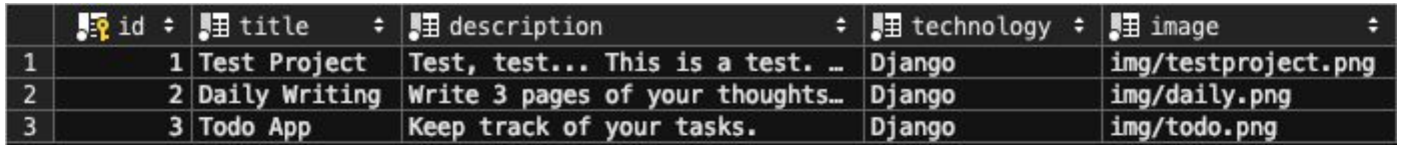


projects_project	
id	integer
title	varchar(100)
description	text
technology	varchar(20)
image	varchar(100)

```python
class Project(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    technology = models.CharField(max_length=20)
    image = models.FilePathField(path="/img")
```



```
projects_project
id            integer
title         varchar(100)
description   text
technology    varchar(20)
image         varchar(100)
```



	id	title	description	technology	image
1	1	Test Project	Test, test... This is a test. …	Django	img/testproject.png
2	2	Daily Writing	Write 3 pages of your thoughts…	Django	img/daily.png
3	3	Todo App	Keep track of your tasks.	Django	img/todo.png

In our app...

PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. Django Models
3. ▶ **Migrations and your SQL Database**
4. Add static files
5. Use the Django Shell
6. Build your routes
7. Create your views
8. Create your templates
9. Add some style!
10. Recap and Outlook

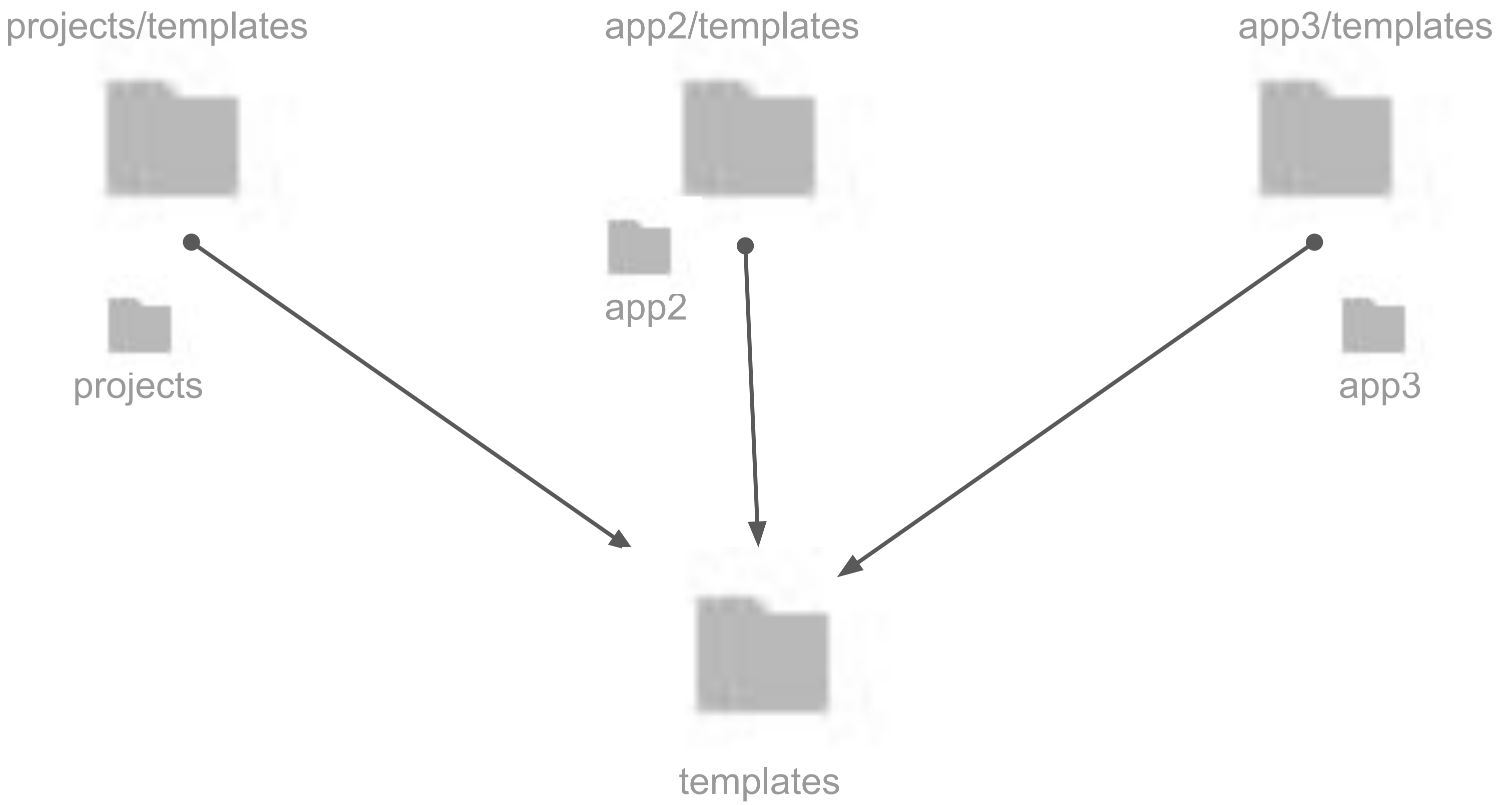
Migrations

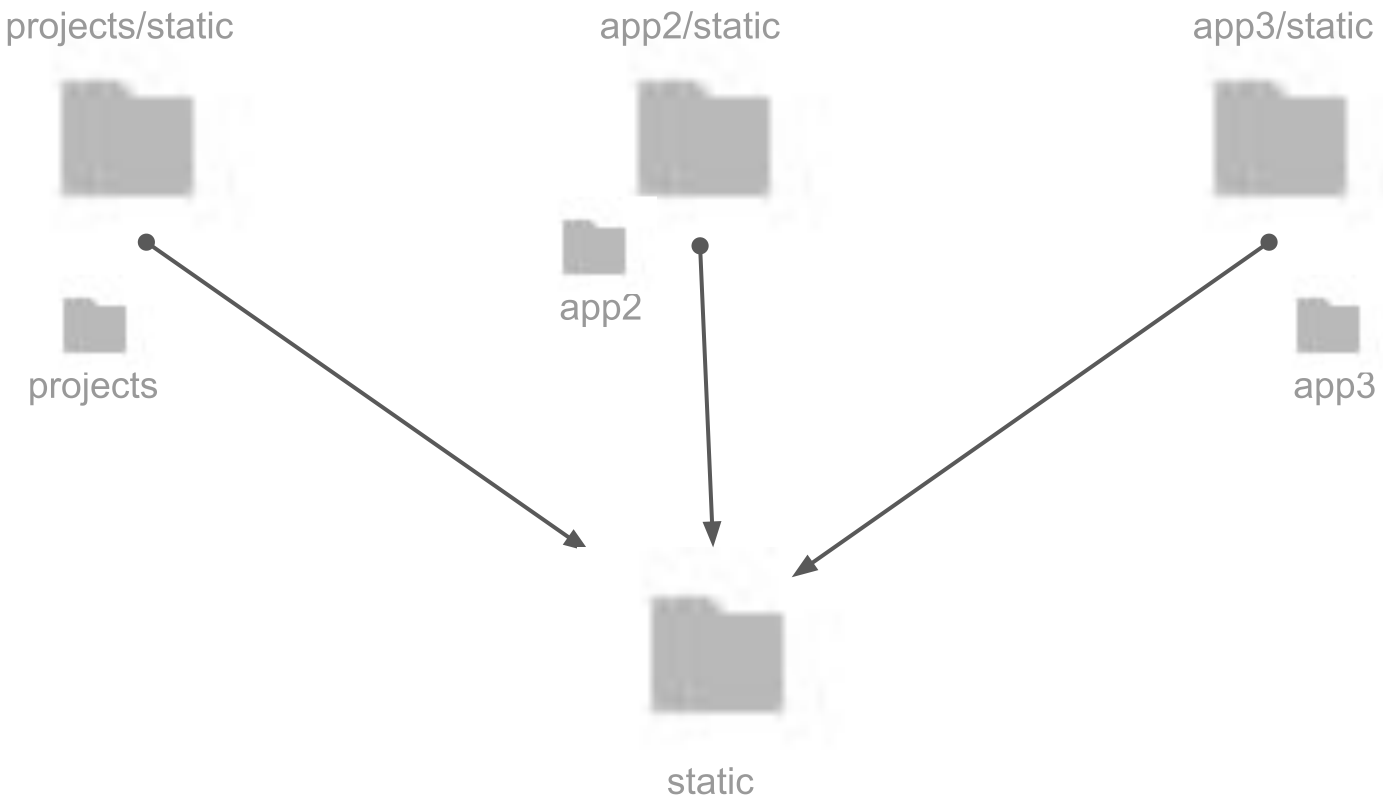
Inside a SQLite file

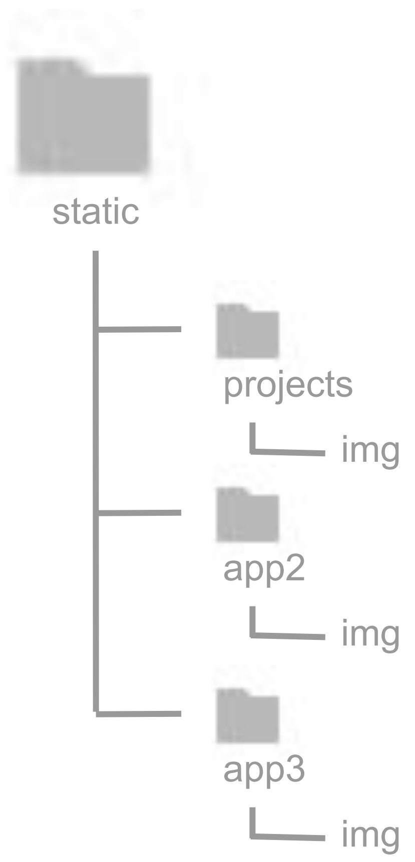
PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. Django Models
3. Migrations and your SQL Database
4. **Add static files**
5. Use the Django Shell
6. Build your routes
7. Create your views
8. Create your templates
9. Add some style!
10. Recap and Outlook

Django’s double-folder structure


projects/templates
app2/templates
app3/templates
app2
projects
app3
templates


projects/static
app2/static
app3/static
app2
projects
app3
static


static
projects
img
app2
img
app3
img

Let’s fix th0t!

Let’s fix th0t!

Let’s fix that!

Let’s fix that!

PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. Django Models
3. Migrations and your SQL Database
4. Add static files
5. **Use the Django Shell**
6. Build your routes
7. Create your views
8. Create your templates
9. Add some style!
10. Recap and Outlook

PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. Django Models
3. Migrations and your SQL Database
4. Add static files
5. Use the Django Shell
6. **Build your routes**
7. Create your views
8. Create your templates
9. Add some style!
10. Recap and Outlook

Error messages are your friends!

PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. Django Models
3. Migrations and your SQL Database
4. Add static files
5. Use the Django Shell
6. Build your routes
7. **Create your views**
8. Create your templates
9. Add some style!
10. Recap and Outlook

Error messages are your friends!

PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. Django Models
3. Migrations and your SQL Database
4. Add static files
5. Use the Django Shell
6. Build your routes
7. Create your views
8. **Create your templates**
9. Add some style!
10. Recap and Outlook

Django Templating Language

{% code logic %}

{{ variables }}

Adding data through the Django Shell

Error messages are your friends!

bugging

ebugging

Debugging

Debugging

Debugging

Debugging

Changing Models and the DB

1) makemigrations

2) migrate

Changing Column Datatypes

Error messages are your friends!

Template Inheritance

PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. Django Models
3. Migrations and your SQL Database
4. Add static files
5. Use the Django Shell
6. Build your routes
7. Create your views
8. Create your templates
9. **Add some style!**
10. Recap and Outlook

PART 4 - SHOWCASE YOUR PROJECTS

1. About this section
2. Django Models
3. Migrations and your SQL Database
4. Add static files
5. Use the Django Shell
6. Build your routes
7. Create your views
8. Create your templates
9. Add some style!
10. **Recap and Outlook**

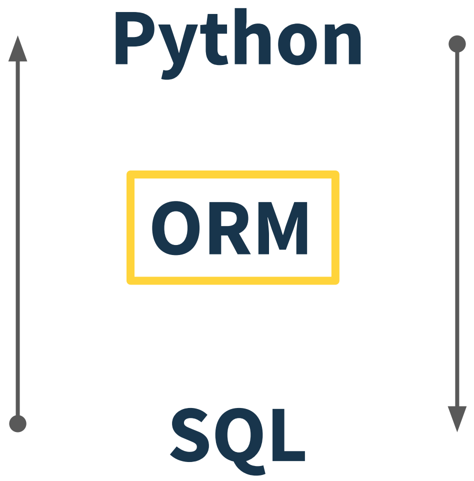
What you did in this section

✓ Created a Django Model

```
class Project(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    technology = models.CharField(max_length=20)
    image = models.CharField(max_length=100)
```

What you did in this section

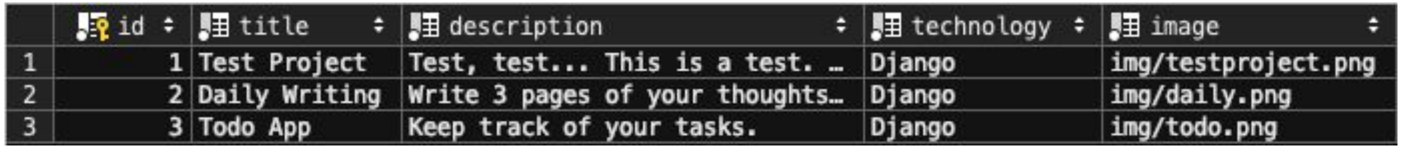
✓ Learned about ORMs (Object-Relational Mappers)



What you did in this section

✓ Learned about ORMs (Object-Relational Mappers)

```python
class Project(models.Model):
    title = models.CharField(max_length=100)
    description = models.TextField()
    technology = models.CharField(max_length=20)
    image = models.CharField(max_length=100)
```



	id	title	description	technology	image
1	1	Test Project	Test, test... This is a test. …	Django	img/testproject.png
2	2	Daily Writing	Write 3 pages of your thoughts…	Django	img/daily.png
3	3	Todo App	Keep track of your tasks.	Django	img/todo.png

What you did in this section

✓ Learned about Relational Databases

Column

Project table

id	title	description	technology	image
1	test	This is a test	Django	/img/t.png
2	daily	Write words	Django	/img/d.png
3	todo	What to do?	Tkinter	/img/td.png

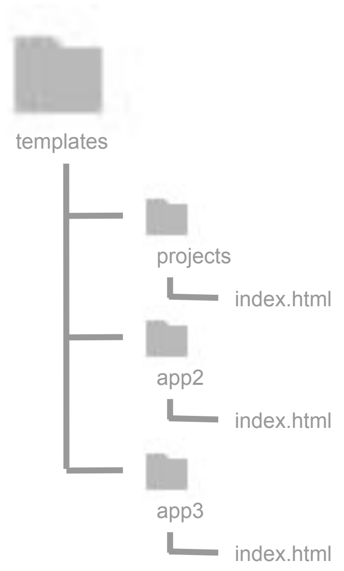
Row

What you did in this section

- ✓ Peeked into a Django Migrations file
- ✓ Peeked into a SQLite Database File

What you did in this section

✓ Revisited the nested template folder structure in Django



What you did in this section

- ✓ Added static image files to your project
- ✓ Linked those image files correctly

```
{% load static %}
```

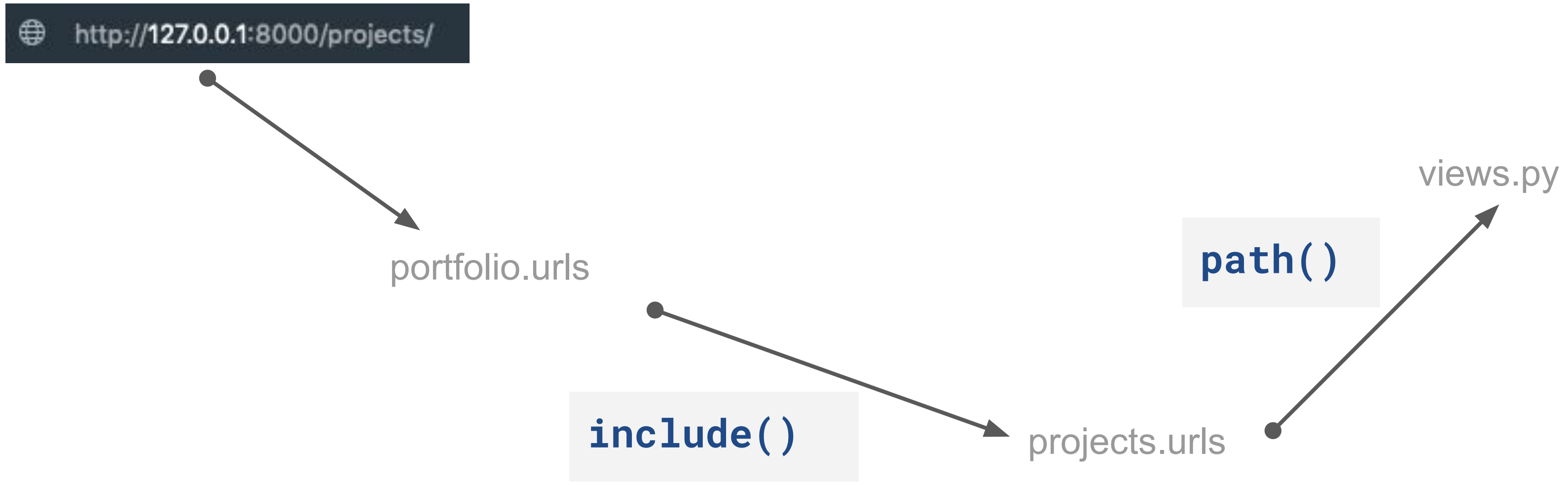
What you did in this section

✓ Used the Django Shell to create and edit data in your DB

```python
projects = Project.objects.all()
p1 = projects[0]
p1.image = "changed/path"
p1.save()
```

What you did in this section

✓ Created new URL paths



What you did in this section

- ✓ Queried the DB from your views
- ✓ Passed data on to your template files

```python
return render(request, "template.html", {"projects": projects})
```

What you did in this section

✓ Used Django Templating Syntax

{% code logic %}

{{ variables }}

What you did in this section

- ✓ Applied more Bootstrap styling

What you did in this section

- ✓ Some serious **debugging**!!!

PART 5 - DISPLAY A SINGLE PROJECT

What you need to continue

- ✓ Set up: Python 3.x, Virtual Environment, Django, text editor
- ✓ Django project “portfolio” with app “projects”
- ✓ Projects in the Database
- ✓ Portfolio page listing all projects

PART 5 - DISPLAY A SINGLE PROJECT

▶ **1. About this section**
2. Django ORM: Accessing a single project
3. Variable arguments from URLs: routing and views
4. URL linking: app_name, path names, and arguments
5. Template Inheritance
6. Some more styling
7. Using the Django Admin Interface
8. Recap and Congrats!
9. Next Steps

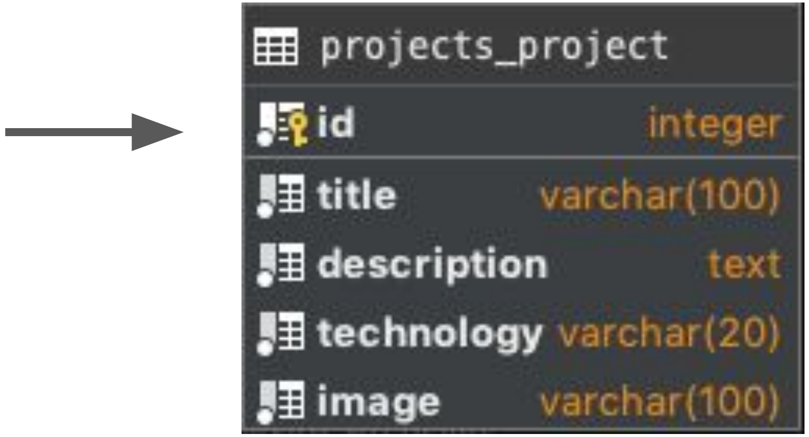
PART 5 - DISPLAY A SINGLE PROJECT

1. About this section
2. **Django ORM: Accessing a single project**
3. Variable arguments from URLs: routing and views
4. URL linking: app_name, path names, and arguments
5. Template Inheritance
6. Some more styling
7. Using the Django Admin Interface
8. Recap and Congrats!
9. Next Steps

Back into the Django Shell

What’s pk?

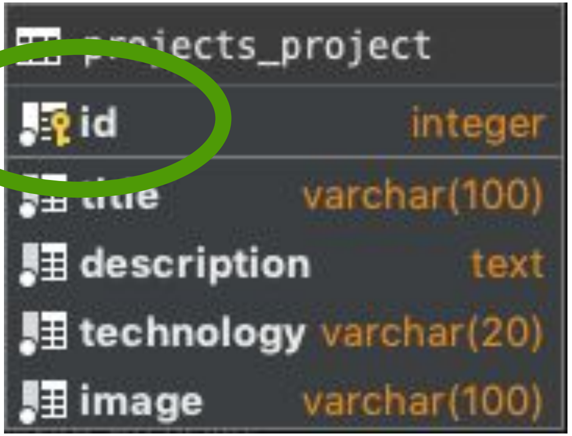
pk means “primary key”



projects_project

id	integer
title	varchar(100)
description	text
technology	varchar(20)
image	varchar(100)

pk means “primary key”



projects_project

id	integer
title	varchar(100)
description	text
technology	varchar(20)
image	varchar(100)

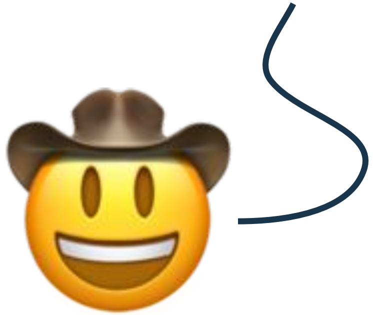
In our project: pk == id

PART 5 - DISPLAY A SINGLE PROJECT

1. About this section
2. Django ORM: Accessing a single project
3. **Variable arguments from URLs: routing and views**
4. URL linking: app_name, path names, and arguments
5. Template Inheritance
6. Some more styling
7. Using the Django Admin Interface
8. Recap and Congrats!
9. Next Steps

Capturing URL parts with < >

Capturing URL parts with < >



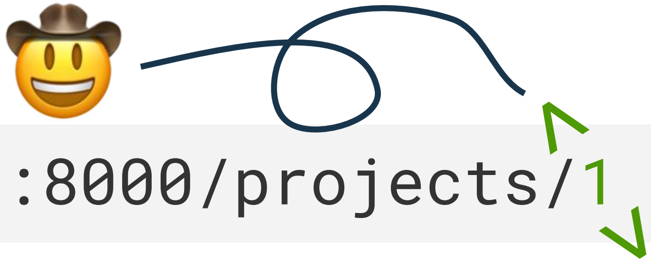
Capturing URL parts

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

```
http://localhost:8000/projects/1
```

Capturing URL parts

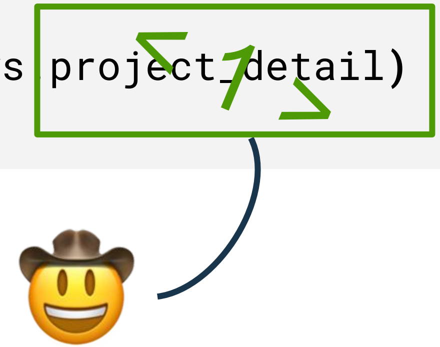
```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```



```
http://localhost:8000/projects/1
```

Capturing URL parts

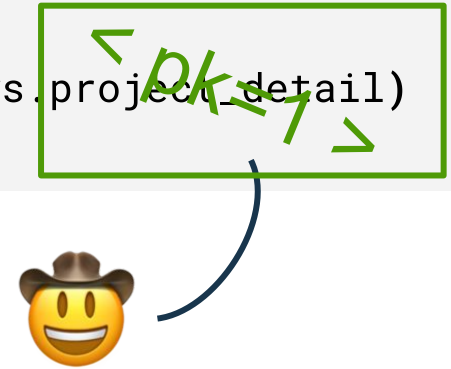
```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```



```
http://localhost:8000/projects/1
```

Capturing URL parts

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```



```
http://localhost:8000/projects/1
```

Path converters - int:

What path converters do

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

What path converters do

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

```
http://localhost:8000/projects/gimmeprojects
```

What path converters do

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

```
http://localhost:8000/projects/gimm    ects
```

What path converters do

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

```
http://localhost:8000/projects/gimm    ects
```

```
http://localhost:8000/projects/1
```

What path converters do

```python
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

```
http://localhost:8000/projects/gimm ✗ ects
```

```
http://localhost:8000/projects/1 ✓
```

Views: the code logic

Resolving URLs

Browser

```
http://localhost:8000/projects/1
```

urls.py

```python
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

views.py

```python
def project_detail(request, pk):
    project = Project.objects.get(pk=pk)
    return render(request,"projects/detail.html", {"projects": project})
```

Resolving URLs

Browser

```
http://localhost:8000/projects/1
```

urls.py

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

views.py

```
def project_detail(request, pk):
    project = Project.objects.get(pk=pk)
    return render(request,"projects/detail.html", {"projects": project})
```

Testing: Can we access a single resource?

PART 5 - DISPLAY A SINGLE PROJECT

1. About this section
2. Django ORM: Accessing a single project
3. Variable arguments from URLs: routing and views
4. ▶ **URL linking: app_name, path names, and arguments**
5. Template Inheritance
6. Some more styling
7. Using the Django Admin Interface
8. Recap and Congrats!
9. Next Steps

Linking two templates

NoReverseMatch

Real Python

Giving names to patterns

Giving names to patterns

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

Giving names to patterns

```
urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

```
<a href="{% url 'app_name:name' project.pk %}">Read More</a>
```

Giving names to patterns

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

```
<a href="{% url 'app_name:name' project.pk %}">Read More</a>
```

Giving names to patterns

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail)
]
```

```
<a href="{% url 'projects:name' project.pk %}">Read More</a>
```

Giving names to patterns

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

```
<a href="{% url 'projects:name' project.pk %}">Read More</a>
```

Giving names to patterns

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Giving names to patterns

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

✓

Identifies Django view

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

NoReverseMatch (but a different one)

Passing arguments for retrieving URLs

Passing arguments for retrieving URLs

your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Passing arguments for retrieving URLs

your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Browser

```
http://localhost:8000/projects/1
```

Passing arguments for retrieving URLs

your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Browser

```
http://localhost:8000/projects/1
```

views.py

```
def project_detail(request, pk):
    project = Project.objects.get(pk=pk)
    return render(request,"projects/detail.html", {"projects": project})
```

🤗

NoReverseMatch

Real Python

Debugging

Check first in these places!

1) **urls.py**

```python
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

Check first in these places!

1) urls.py

```python
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

Check first in these places!

1) urls.py

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

Check first in these places!

2) your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Check first in these places!

2) your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

NoReverseMatch Debugging 1: Identify view

urls.py

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Check first in these places!

your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Check first in these places!

your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Browser

```
http://localhost:8000/projects/1
```

Check first in these places!

your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Browser

```
http://localhost:8000/projects/1
```

views.py

```
def project_detail(request, pk):
    project = Project.objects.get(pk=pk)
    return render(request,"projects/detail.html", {"projects": project})
```

NoReverseMatch Debugging 2: Pass arguments

urls.py

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

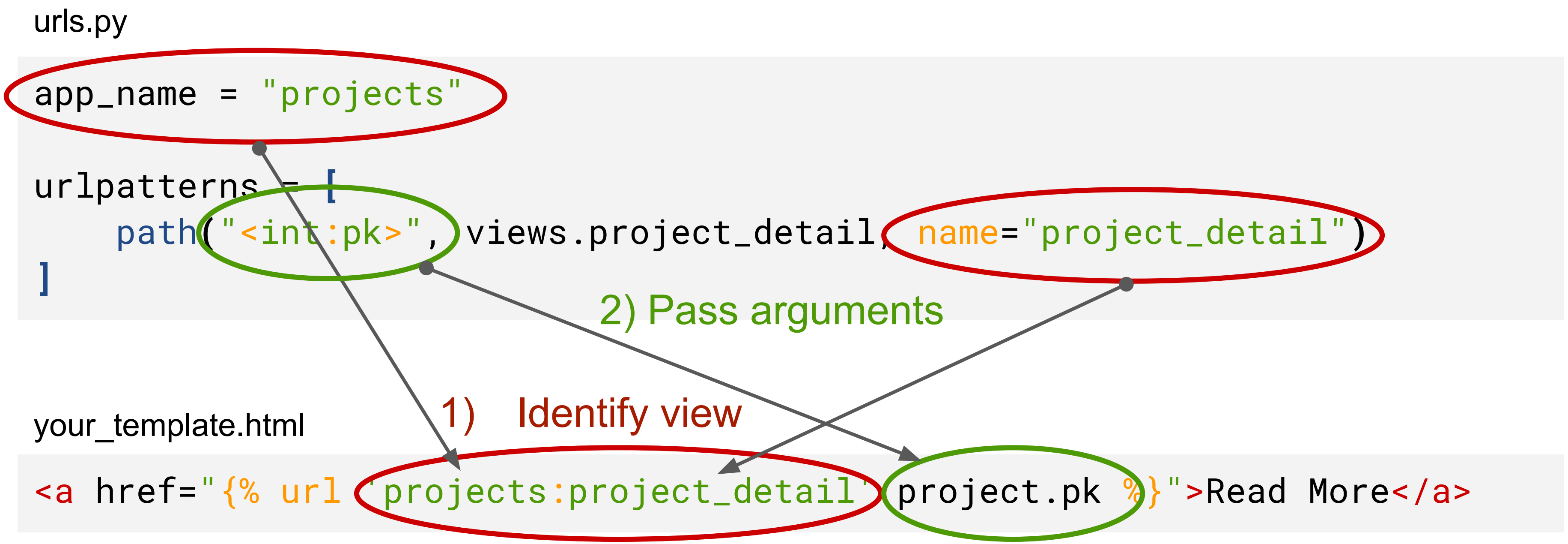
your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

Debugging Summary

Debugging Summary

NoReverseMatch Debugging Summary



urls.py

```
app_name = "projects"

urlpatterns = [
    path("<int:pk>", views.project_detail, name="project_detail")
]
```

your_template.html

```
<a href="{% url 'projects:project_detail' project.pk %}">Read More</a>
```

PART 5 - DISPLAY A SINGLE PROJECT

1. About this section
2. Django ORM: Accessing a single project
3. Variable arguments from URLs: routing and views
4. URL linking: app_name, path names, and arguments
5. **Template Inheritance**
6. Some more styling
7. Using the Django Admin Interface
8. Recap and Congrats!
9. Next Steps

DRY - don’t repeat yourself

File 1: base.html

base.html

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

base.html

```
{% block content %}

{% endblock %}
```

base.html

```
{% block content %}




{% endblock %}
```

base.html

```
{% block content %}

<div>My HTML content</div>

{% endblock %}
```

File 2: your_template.html

your_template.html

```
{% extends ‘base.html’ %}

{% block content %}



{% endblock %}
```

your_template.html

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

{% block content %}



{% endblock %}
```

your_template.html

```
{% extends ‘base.html’ %}

{% block content %}

<div>My HTML content</div>

{% endblock %}
```

In practice!

PART 5 - DISPLAY A SINGLE PROJECT

1. About this section
2. Django ORM: Accessing a single project
3. Variable arguments from URLs: routing and views
4. URL linking: app_name, path names, and arguments
5. Template Inheritance
6. **Some more styling**
7. Using the Django Admin Interface
8. Recap and Congrats!
9. Next Steps

Mobile responsive!

```html
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

PART 5 - DISPLAY A SINGLE PROJECT

1. About this section
2. Django ORM: Accessing a single project
3. Variable arguments from URLs: routing and views
4. URL linking: app_name, path names, and arguments
5. Template Inheritance
6. Some more styling
7. **Using the Django Admin Interface**
8. Recap and Congrats!
9. Next Steps

A convenient way of accessing your DB

Creating a superuser

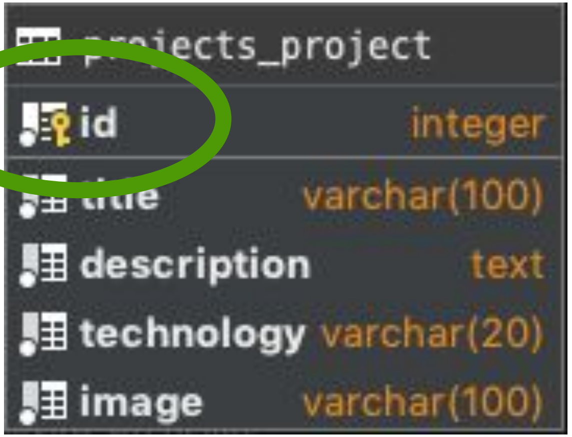
Registering your models

PART 5 - DISPLAY A SINGLE PROJECT

1. About this section
2. Django ORM: Accessing a single project
3. Variable arguments from URLs: routing and views
4. URL linking: app_name, path names, and arguments
5. Template Inheritance
6. Some more styling
7. Using the Django Admin Interface
8. **Recap and Congrats!**
9. Next Steps

What you did in this section

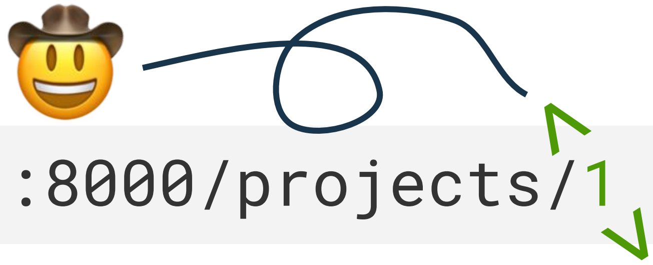
- ✓ Using the Django ORM in the shell and in views.py
- ✓ Learned about primary keys (pk)



projects_project	
id	integer
title	varchar(100)
description	text
technology	varchar(20)
image	varchar(100)

What you did in this section

- ✓ Lasso-ed parts from URLs using angle brackets < >



```
http://localhost:8000/projects/1
```

What you did in this section

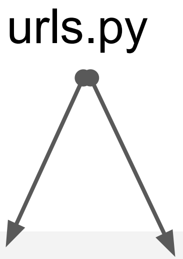
✓ Used an int: path converter to verify the URL path's datatype

```
http://localhost:8000/projects/gimm ects
```

```
http://localhost:8000/projects/1
```

What you did in this section

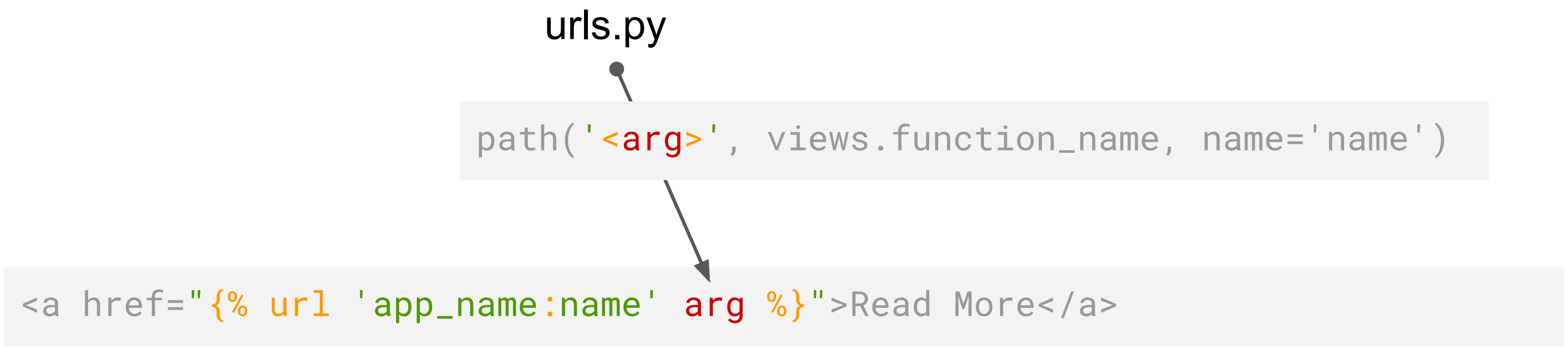
✓ Defining proper namespaces for linking and reverse URL matching



```
<a href="{% url 'app_name:name' %}">Read More</a>
```

What you did in this section

✓ Passing arguments inside of URL template tags



```
path('<arg>', views.function_name, name='name')
```

```
<a href="{% url 'app_name:name' arg %}">Read More</a>
```

What you did in this section

✓ Debugging NoReverseMatch Error

urls.py

- Identify view
- Pass arguments

your_template.html

What you did in this section

- ✓ DRY - don’t repeat yourself
- ✓ Template inheritance

```
{% extends ‘base.html’ %}

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

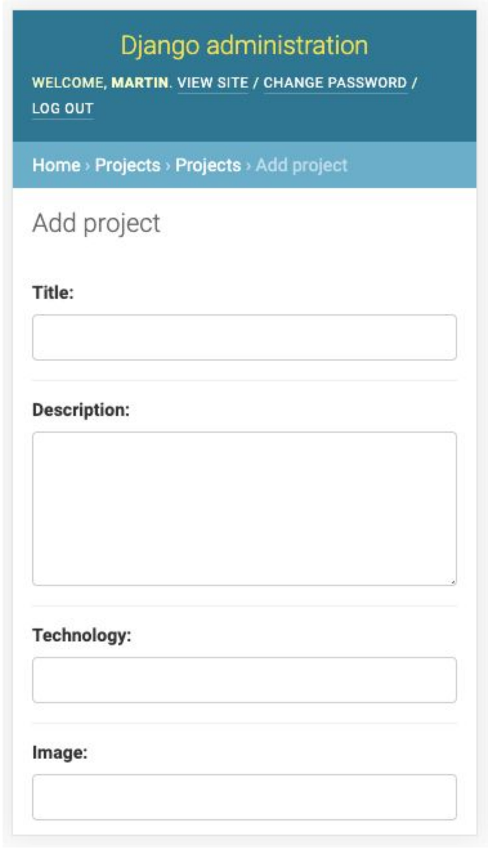
What you did in this section

- ✓ Bootstrap header
- ✓ Make the site mobile responsive

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

What you did in this section

- ✓ Create a superuser
- ✓ Register your models in Django admin
- ✓ Add/Edit/Delete DB entries conveniently


Django administration
WELCOME, MARTIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Projects › Projects › Add project
Add project
Title:
Description:
Technology:
Image:

CONGRATULATIONS!!

What you built

django

Share your knowledge with a Blog

🐍 **Keep learning!** 🐍