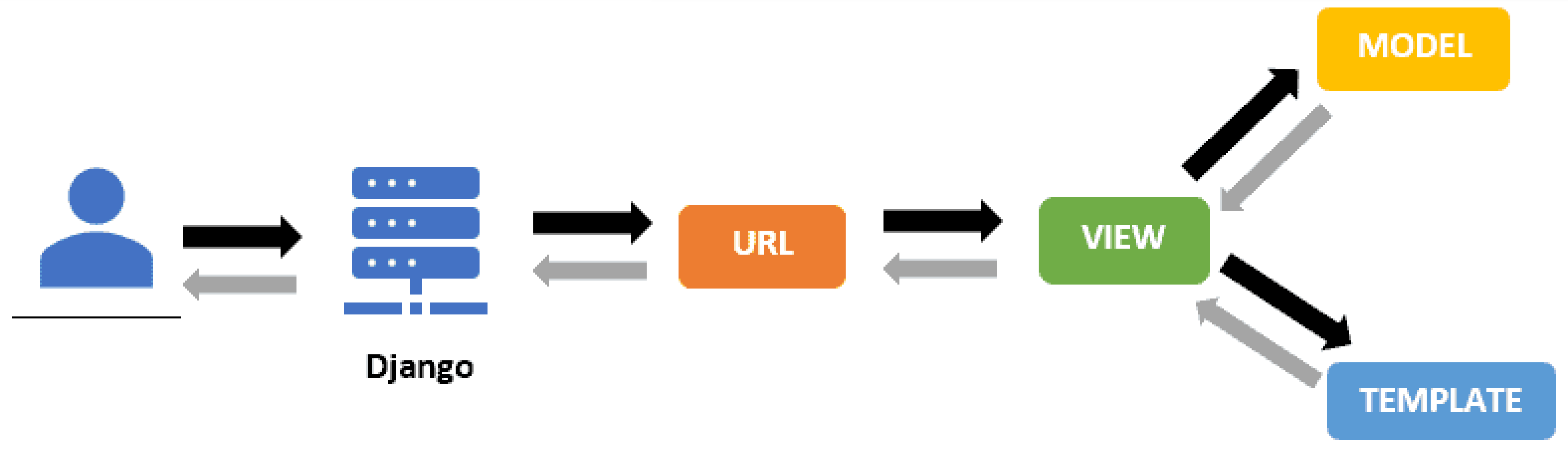
Django Urls and Views

Web Programming 2

Dr Aaron Mininger

29 Feb 2024

Django Framework



- A **View** is a function that takes a **request** as input and returns a **response**
- A **Model** represents a table in the database
- A **Template** is an HTML document with additional code to help render it
- A **Url** is a mapping from a path to a view

Course Project

In this course, we will be designing a food blog with recipes.

Getting Started

Step 1: Create a virtual environment

A python **virtual environment** is an environment where you can install packages locally, to reduce conflicts between different projects

Getting Started

Create a directory for your project:

```
$ mkdir chakula_chat
$ cd chakula_chat
```

Virtual Environments

Create a virtual environment

```
$ python -m venv cc_venv
```

Virtual Environments

Source (load) the virtual environment

```
$ source cc_venv/bin/activate
```

Virtual Environments

Install python packages with pip:

```
$ pip install django

# (If using MySQL)
$ pip install mysqlclient
```

Create the Django Project

Step 2: Create the Django Project

```
$ django-admin startproject chakula_chat
```

Create the Django Project

The `django-admin` tool creates the following structure:

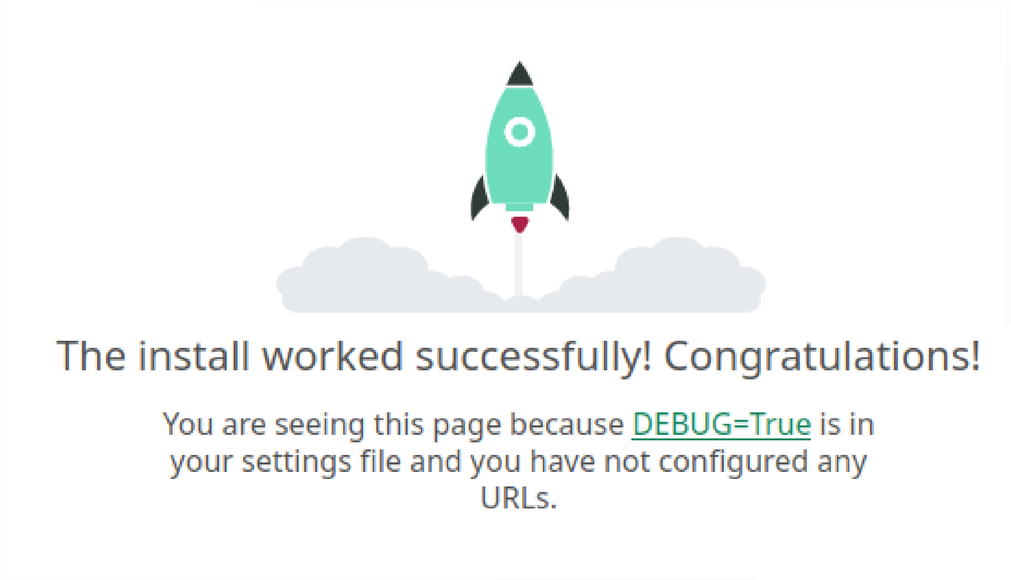
```
chakula_chat/
    manage.py          # Command line scripts for managing the project
    chakula_chat/      # The main python package for your app
        settings.py    # A file with project settings/configuration
        urls.py        # The root url configuration file
```

Testing the Project

In the root project directory, run your project locally with:

```
$ python manage.py runserver
```

Then, in your web browser, go to `http://127.0.0.1:8000`



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 Apps

A Django project should have one or more **apps**

An app is a directory/package with its own urls/views/models

Think of it as a specific part/feature of your website

Django Apps

In our recipe blog, we will create an app for certain core pages

To create an app, do the following:

```
$ python manage.py startapp pages
```

Django Apps

Django will create the following files:

```
chakula_chat/
    manage.py
    chakula_chat/
    pages/
        admin.py    # Manage the admin page for this app
        apps.py     # Root python file to load this app
        models.py   # Defines the app's object models
        tests.py    # Place to write your own tests
        views.py    # Defines the app's views
```

Django Apps

!!! Important !!!

Add the app to the list of `INSTALLED_APPS` in our `settings.py` file:

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

Serving a Static Webpage

Our first task: serve a static webpage (html file)

Need the following:

1. An html file
2. A view
3. A url

Lets create a home page for our site

Creating html file

Html files should go inside a directory called `APP_DIR/templates/APP_NAME/`

Example:

Create a file `pages/templates/pages/home.html` as our home page

Creating a view

Next, we will create a view for this home page

Example:

Open the file `pages/views.py` and write:

```python
from django.shortcuts import render

def home(request):
    return render(request, 'pages/home.html')
```

Creating a view

A **view** is a function that

- Takes as input an `HttpRequest` object
- Returns an `HttpResponse` object

```python
from django.shortcuts import render

def home(request):
    return render(request, 'pages/home.html')
```

Creating a view

Here our view is named `home`

- Takes a parameter `request` which is of type `HttpRequest`
- Returns an `HttpResponse` using the `render` shortcut function

```python
from django.shortcuts import render

def home(request):
    return render(request, 'pages/home.html')
```

Render Function

Function `django.shortcuts.render`

`render(request: HttpRequest, template_file: str) -> HttpResponse`

- Always pass the request as the first parameter
- The second parameter is the name of the template file

```python
from django.shortcuts import render

def home(request):
    return render(request, 'pages/home.html')
```

Creating a url

Finally, we need to create a mapping from a url to a view

Example:

Open `chakula_chat/urls.py`

```
from django.contrib import admin
from django.urls import path

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

Urls

The file `urls.py` must contain a list called `urlpatterns`

This will be a list of `path` objects

```
from django.contrib import admin
from django.urls import path

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

Urls

Function `django.urls.path`

```
path(route: str, view_fn: function)
```

Urls

Function `django.urls.path`

```
path(route: str, view_fn: function)
```

The url **route** is everything after the website name:

url	route
`http://127.0.0.1/`	`''`
`http://127.0.0.1/pages/`	`'/pages/'`
`http://127.0.0.1/recipes/view/3142/`	`'/recipes/view/3142/'`

Urls

For a single view, we add a new `path(route, view)` entry to the `urlpatterns`

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

from pages import views as page_views      # NEW

urlpatterns = [
    path('', page_views.home),             # NEW

    path('admin/', admin.site.urls),
]
```

Demo

Now, if we go to `http://127.0.0.1:8000/` we will see our html page

Summary

1. Create an html file in `APP_DIR/templates/APP_NAME/`

2. Create a view in `APP_DIR/views.py`

```python
def view_name(request):
    return render(request, 'APP_NAME/page.html')
```

3. Create a url mapping:

```python
urlmappings = [
    path('/path/to/page/', view_fn)
]
```

More Pages

Let's add some more pages (about, contact)

1. Create the html files
2. Create the views
3. Create the url mappings

More Page

About Page

File	Content
`pages/templates/pages/about.html`	Html File
`pages/views.py`	`def about(request):`
`chakula_chat/urls.py`	`path('about/', page_views.about)`

Contact Page

File	Content
`pages/templates/pages/contact.html`	Html File
`pages/views.py`	`def contact(request):`
`chakula_chat/urls.py`	`path('contact/', page_views.contact)`

More Page

Now we can go to `http://127.0.0.1:8000/contact/` and `http://127.0.0.1:8000/about/`

Hyperlinks

Now, suppose we want to add links from our home page to our new pages:

```html
<!-- home.html -->
<a href="/about/">About Me</a>
<a href="/contact/">Contact Me</a>
```

Hyperlinks

Now, suppose we want to add links from our home page to our new pages:

```html
<!-- home.html -->
<a href="/about/">About Me</a>
<a href="/contact/">Contact Me</a>
```

Note: It is generally not good practice to write urls like this

Why? If the link changes, you have to change it in all your files

Hyperlinks

Better practice for urls:

Give each url a name and refer to its name instead of path

```
# urls.py
urlmapping = [
    path('', page_views.home, name='home'),
    path('contact/', page_views.contact, name='contact'),
    path('about/', page_views.about, name='about'),
]
```

Hyperlinks

After we give a url path a name,

we can use it in our html templates:

```
<!-- home.html -->
<a href="{% url 'about' %}">About Me</a>
<a href="{% url 'contact' %}">Contact Me</a>
```

Hyperlinks

This code `{% url 'url_name' %}` is a Django **template tag**

When Django **renders** a template, it will replace the tags with text

This allows Django to run code when generating html files

Hyperlinks

This code `{% url 'url_name' %}` is a Django **template tag**

When Django **renders** a template, it will replace the tags with text

This allows Django to run code when generating html files

Django Template:

```
<!-- home.html -->
<a href="{% url 'about' %}">About Me</a>
```

Sent to User:

```
<a href="/about/">About Me</a>
```

Url Names

Summary: Using urls in files

`urls.py`

Write: `path('/url/route/', view_fn, name='url_name')`

`page.html`

Write: `<a href="{% url 'url_name' %}">`

Including Urls

Currently we are defining the pages urls in the root project:

```python
# chakula_chat/urls.py

urlpatterns = [
    # Standalone Web Pages
    path('', page_views.home, name='home'),
    path('about/', page_views.about, name='about'),
    path('contact/', page_views.contact, name='contact'),

    path('admin/', admin.site.urls),
]
```

Including Urls

It is generally better to define each app's urls locally

Example:

Create a new file `pages/urls.py` and write the following:

```python
# pages/urls.py

from django.urls import path
from . import views

urlpatterns = [
    # Standalone Web Pages
    path('', views.home, name='home'),
    path('about-me/', views.about, name='about'),
    path('contact/', views.contact, name='contact'),
]
```

Including Urls

Then, we can just include this file in our root project urls.py

Example:

Change the file `chakula_chat/urls.py` to the following:

```python
# chakula_chat/urls.py

from django.contrib import admin
from django.urls import path, include  # Add import for include

urlpatterns = [
    # Standalone Web Pages
    path('', include('pages.urls')),

    path('admin/', admin.site.urls),
]
```

Namespacing Urls

As your site grows and has many urls, it can be good practice to put the urls in a namespace

```python
# pages/urls.py

from django.urls import path
from . import views

app_name = "pages"   # Defines the namespace
urlpatterns = [
    path('', views.home, name='home'),
    path('about-me/', views.about, name='about'),
    path('contact/', views.contact, name='contact'),
]
```

Then in your templates: `<a href="{% url 'pages:home' %}>Home</a>`

Static Files

What if we want to include **static files** to our page?

- CSS, JavaScript, Images, etc.

Static Files

What if we want to include **static files** to our page?

- CSS, JavaScript, Images, etc.

Note: some static files are app specific, and some are used across the project

(such as a base css theme)

Static Files

We are going to create a new app called `base` which will just hold static files and templates used across the whole site

```
$ python manage.py startapp base
```

Remember to edit the settings file:

```
# settings.py
INSTALLED_APPS = [
    'base',
    'pages',
    ...
]
```

Static Files

To set up static files, edit the settings.py file:

```
# settings.py

STATIC_URL = 'static/'
STATIC_ROOT = BASE_DIR / 'staticfiles'
```

Static Files

Then, edit your main `urls.py` file

```python
# chakula_chat/urls.py
from django.contrib import admin
from django.urls import path, include
from django.conf import settings                # NEW
from django.conf.urls.static import static    # NEW

urlpatterns = [
    # Standalone Web Pages
    path('', include('pages.urls')),

    path('admin/', admin.site.urls),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT) # NEW
```

Static Files

Let's create a base css file to use across the site

Create a new directory `base/static/`

Then create a directory inside `css/` and add a file `base.css`

```css
/** base.css **/

h1 {
    color: red;
}
```

Static Files

Let's include this css file in our `home.html` file

```html
<!-- home.html -->

<head>
    <link rel="stylesheet" href="/static/css/base.css" type="text/css">
</head>
```

Now our title is red!

Static Files

Note: like normal url's, there is a preferred way to link to static files:

```
<!-- home.html -->
{% load static %}

<head>
    <link rel="stylesheet" href="{% static 'css/base.css' %}" type="text/css">
</head>
```

Note: You **must** add the `{% load static %}` tag at the top

Static Files

Now lets add a new banner image to our home page.

Because this is specific to a certain page, we will add it to the pages app

Note: I recommend the following static file structure:

```
pages/
    static/
        pages/
            css/
            js/
            img/
```

Static Files

Now lets add a new banner image to our home page.

Because this is specific to a certain page, we will add it to the pages app

Let's copy the image to `pages/static/pages/img/food_banner.jpg`

Then in our html file:

```
<!-- home.html -->

<img src="{% static 'pages/img/food_banner.jpg' %}">
```

Summary - Project Structure

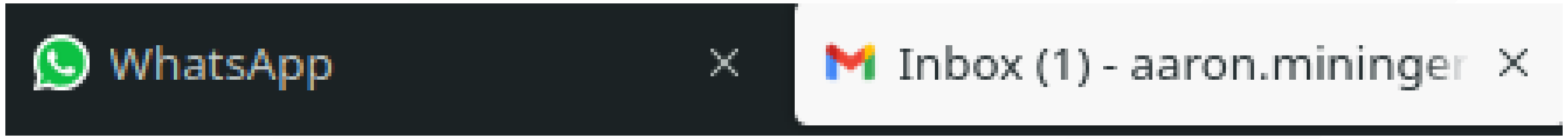
In the end, here is our project structure:

```
chakula_chat/
    manage.py
    chakula_chat/
        settings.py
        urls.py
    base/
        static/
            css/
                base.css
    pages/
        urls.py
        views.py
        static/
            pages/
                img/
                    food_banner.jpg
        templates/
            pages/
                home.html
                contact.html
                about.html
```

Favicon

Let's add a **favicon** to our site

This is the little icon that goes beside the url at the top



Favicon

First, add the image to our static files:

`base/static/img/favicon.png`

Then, add the link to our html head:

```html
<!-- home.html -->
<head>
    <link rel="icon" href="{% static 'img/favicon.png' %}">

</head>
```

Templates

Let us consider our html pages:

```
<!-- home.html -->
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta name="description" content="Blog about Congolese food">
    <meta name="author" content="Aaron Mininger">

    <title>Chakula Chat Food Blog</title>

    <link rel="icon" href="{% static 'img/favicon.png' %}">
    <link rel="stylesheet" href="{% static 'css/base.css' %}" type="text/css">
</head>

<body>
    <h1>Chakula Chat Food Blog</h1>
    <p><em>Welcome!</em></p>
    <p><a href="{% url 'about' %}">About Me</a></p>
    <p><a href="{% url 'contact' %}">Contact Me</a></p>
</body>
</html>
```

Templates

Notice that most of this would be the same for every page

```html
<!-- home.html -->
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta name="description" content="Blog about Congolese food">
    <meta name="author" content="Aaron Mininger">

    <title>Chakula Chat Food Blog</title>

    <link rel="icon" href="{% static 'img/favicon.png' %}">
    <link rel="stylesheet" href="{% static 'css/base.css' %}" type="text/css">
</head>

<body>
    <h1>Chakula Chat Food Blog</h1>
    <p><em>Welcome!</em></p>
    <p><a href="{% url 'about' %}">About Me</a></p>
    <p><a href="{% url 'contact' %}">Contact Me</a></p>
</body>
</html>
```

Templates

In Django, we can have html templates extend or include other ones

Let's create a new file `base/templates/base-template.html`

```html
<!-- base-template.html -->
{% load static %}
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <meta name="description" content="Blog about Congolese food">
    <meta name="author" content="Aaron Mininger">

    <title>Chakula Chat Food Blog</title>

    <link rel="icon" href="{% static 'img/favicon.png' %}">
    <link rel="stylesheet" href="{% static 'css/base.css' %}" type="text/css">
</head>

<body>
    {% block 'content' %}
    {% endblock %}
```

Templates

Now, our `home.html` file can extend this template:

This lets us avoid repeat code

```
<!-- home.html -->
{% extends 'base-template.html' %}
{% load static %}

{% block 'content' %}
    <h1>Chakula Chat Food Blog</h1>
    <p><em>Welcome!</em></p>
    <p><a href="{% url 'about' %}">About Me</a></p>
    <p><a href="{% url 'contact' %}">Contact Me</a></p>
{% endblock %}
```

Templates

Now, our `home.html` file can extend this template:

The tag `{% extends 'base-template.html' %}` says that this page will be created from the given template

```
<!-- home.html -->
{% extends 'base-template.html' %}
{% load static %}

{% block 'content' %}
    <h1>Chakula Chat Food Blog</h1>
    <p><em>Welcome!</em></p>
    <p><a href="{% url 'about' %}">About Me</a></p>
    <p><a href="{% url 'contact' %}">Contact Me</a></p>
{% endblock %}
```

Templates

Now, our `home.html` file can extend this template:

Then the `{% block 'content' %}` and `{% endblock %}` section will replace it in the original template

```
<!-- home.html -->
{% extends 'base-template.html' %}
{% load static %}

{% block 'content' %}
    <h1>Chakula Chat Food Blog</h1>
    <p><em>Welcome!</em></p>
    <p><a href="{% url 'about' %}">About Me</a></p>
    <p><a href="{% url 'contact' %}">Contact Me</a></p>
{% endblock %}
```

Templates

We can modify the other pages too:

```
<!-- about.html -->
{% extends 'base-template.html' %}
{% load static %}

{% block 'content' %}
    <h1>About Me</h1>

    <p>
        My name is Aaron Mininger, and I love to cook!
        I have been living in Congo for the past 2 years
        and have collected several recipes in my time
        here that I want to share with you.
    </p>
{% endblock %}
```

Templates

We should also do this for the title, since it is different on each page:

```
<!-- base-template.html -->
<head>
    <title>{% block 'title' %} {% endblock %}</title>
</head>
```

```
<!-- home.html -->
{% extends 'base-template.html %}

{% block 'title' %}Chakula Chat Food Blog{% endblock %}
```