

# 3 - Pages app

01 September 2018 21:36

D:\nchaurasia\Python-Architect\Django-William-Vincent\pages

```
$ mkdir pages
$ cd pages
$ pipenv install django==2.0.6
$ pipenv shell
(pages) $ django-admin startproject pages_project .
(pages) $ python manage.py startapp pages
```

```
□ # pages_project/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'pages', # new
]
```

```
□ (pages) $ python manage.py runserver
```

## Templates

**Inside:** D:\nchaurasia\Python-Architect\Django-William-Vincent\pages\pages

```
|— templates
  |— pages
    |— home.html
```

- This means we would need to create a new templates directory, a new directory with the name of the app, pages, and finally our template itself which is home.html.
- there's another often-used approach to structuring the templates in a Django project. And that is to instead create a single, project-level templates directory that is available to all apps. This is the approach we'll use. By making a small tweak to our settings.py file we can tell Django to also look in this project-level folder for templates.
- (pages) \$ mkdir templates  
(pages) \$ touch templates/home.html
- Next we need to update settings.py to tell Django to look at the project-level for templates. This is a one-line change to the setting 'DIRS' under TEMPLATES.

- # pages\_project/settings.py

```
TEMPLATES = [  
  
    'DIRS': [os.path.join(BASE_DIR, 'templates')], #new  
  
]
```

- <!-- templates/home.html -->  
<h1>Homepage.</h1>

## Class-Based Views

- Function-based generic views were introduced to abstract these patterns and streamline development of common patterns.
- However there was no easy way to extend or customize these views.
- As a result, Django introduced class-based generic views that make it easy to use and also extend views covering common use cases.
- **Django-William-Vincent\pages\pages\views.py**

```
from django.views.generic import TemplateView  
  
class HomePageView(TemplateView):  
    template_name = 'home.html'
```

## URLs

```
# pages_project/urls.py  
from django.contrib import admin  
from django.urls import path, include
```

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

- Next create an app-level urls.py file.

```
# pages/urls.py  
from django.urls import path  
from . import views  
  
urlpatterns = [  
    path("", views.HomePageView.as_view(), name='home'),  
]
```

- When using Class-Based Views, you always add as\_view() at the end of the view name.

Python manage.py runserver

## Add an About Page

- **new template file,**
- **a new view, and**
- **a new url route.**
  
- **new template file,**  
Django-William-Vincent\pages\templates\about.html
- **a new view, and**  
Django-William-Vincent\pages\pages\views.py

```
class AboutPageView(TemplateView):  
    template_name = 'about.html'
```

- **a new url route.**  
Django-William-Vincent\pages\pages\urls.py

```
path('about/', views.AboutPageView.as_view(), name='about'),
```

Start server here....

## Extends Templates

- Extending Templates The real power of templates is their ability to be extended.
- If you think about most web sites, there is content that is repeated on every page (header, footer, etc).

```
(pages) $ touch templates/base.html  
<!-- templates/base.html -->
```

■ ■ ■ ■

```
(pages) $ touch templates/base.html
```

```
<header>  
  <a href="{% url 'home' %}">Home</a> | <a href="{% url 'about' %}">About</a>  
</header>
```

```
{% block content %}  
<h1>I am base file.</h1>  
{% endblock %}
```

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

```
<!-- templates/about.html -->
```

