universidade de aveiro
departamento de electrónica, telecomunicações e informática

Technologies and Web Programming

Introduction to the Django Platform

Introduction

- Django is a free and open source platform, written in Python, for developing web applications.
- Named after a famous guitarist “Django Reinhardt”
- It is maintained by the Django Software Foundation (DSF), an independent organization
- Fosters rapid, clean and pragmatic development
- Created in 2003, it became open source in 2005

Features

- Partially follows the MVC pattern
- It has an ORM *(Object Relational Mapper)* to process data
- Focused on automation, adhering to the DRY principle (Don't Repeat Yourself)
- Uses a template system
- Admin customization system, to facilitate CRUD
- Elegant URL routing design
- Has a built-in light web server (for testing)
- Enables the use of customized middleware
- Has facilities for: authentication, internationalization and caching

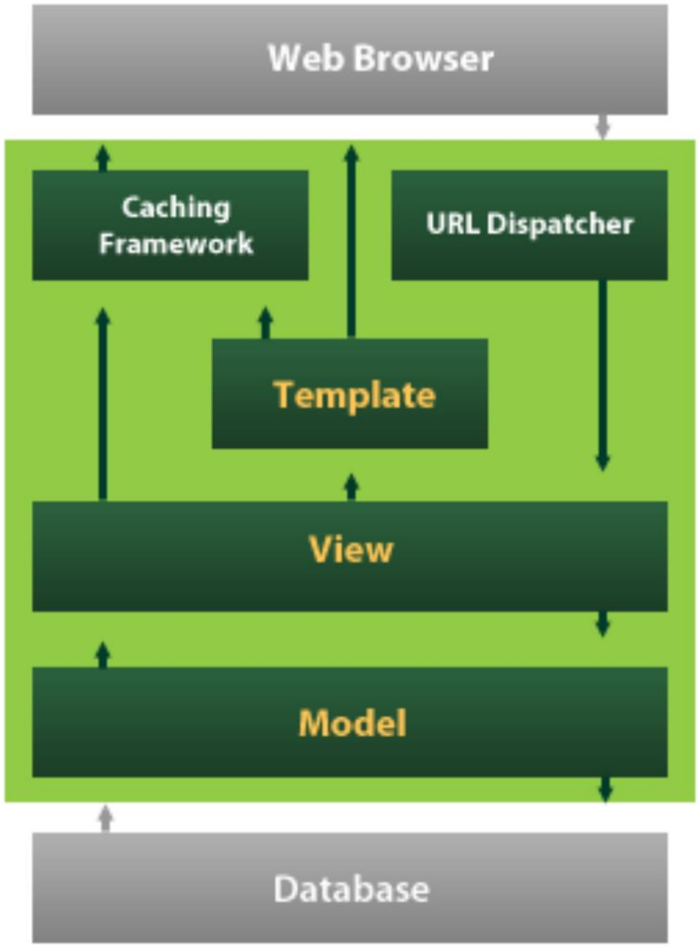
Architecture

Models Describe the data

Views Control what users see

Control **Templates** as they come

URL Dispatcher URL Dispatcher



Django Project Structure

webproj/ ------ Folder for the project. It can have any name.

manage.py -- Command-line utility to interact with the project. **webproj/** --- Project package. Name used for imports.

__init__.py --- File that defines this folder as a package, in Python. settings.py --- Django project settings. urls.py ------- Mapping/routing the URLs for this project. wsgi.py ------- An entry point for WSGI-compatible *webservers .*

app/ ------- Individual web application, with the possibility of several coexisting. templates/ ---- HTML files, invoked by views. static/ ------- CSS, JS, images, etc. – configurable in “settings.py” __init__.py -- File that defines this folder as a package, in Python. views.py ------ Receives requests from customers and returns responses. models.py ----- Data models. admin.py ------ Automatic creation of interface for the data model. forms.py ------ Allows the reception of data sent by clients.

Settings

l The Django project's settings.py file overrides the <python>/Lib/ sitepackages/django/conf/global_settings.py file

l Attributes:

l DEBUG # True or False

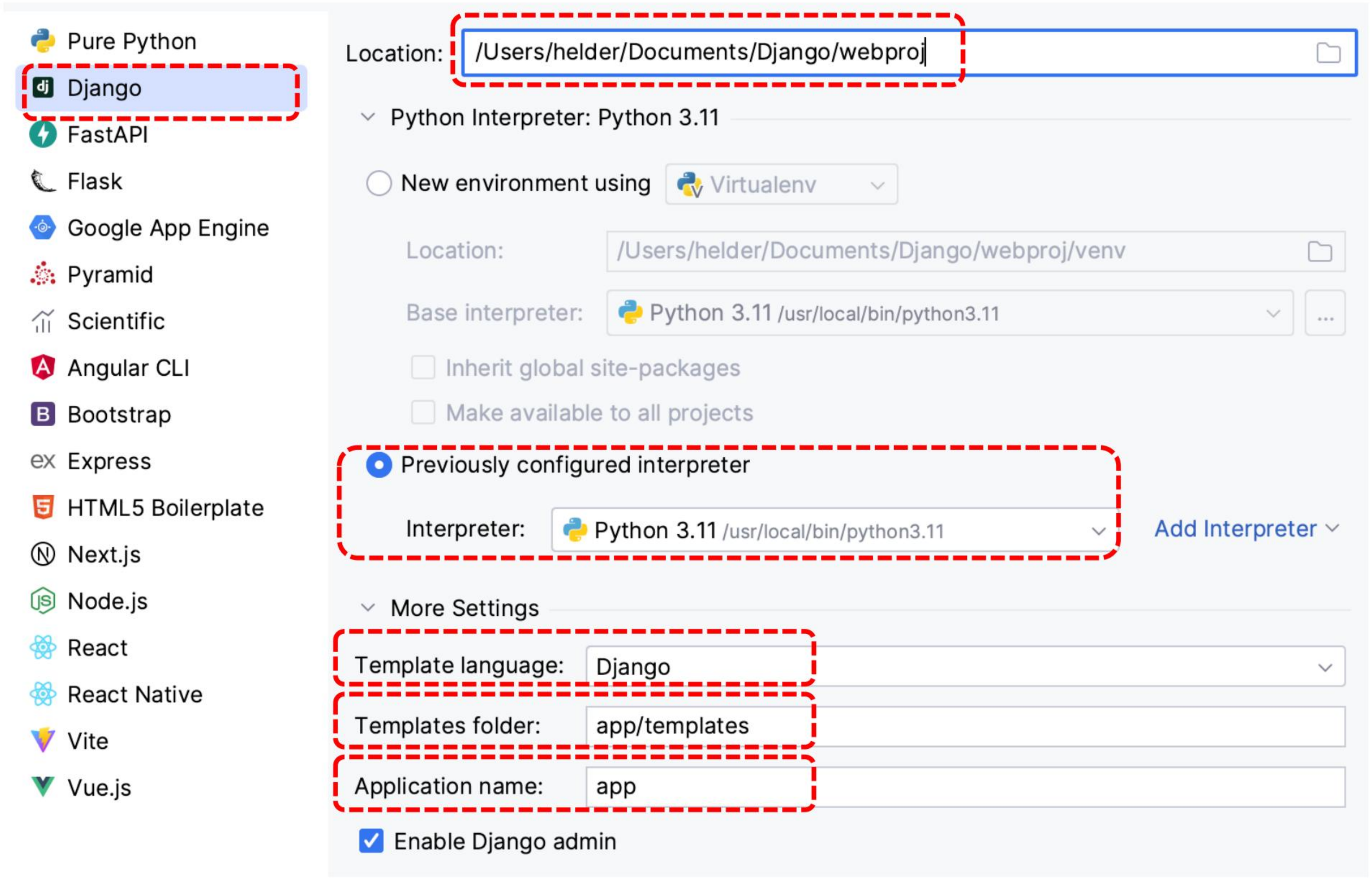
l DATABASES ENGINE # 'mysql', 'sqlite3', 'oracle' ... etc. l
ROOT_URLCONF # URL routing configuration l
MEDIA_ROOT # For user-uploaded files l MEDIA_URL # For multimedia files
l STATIC_ROOT # Folder for static files such
as CSS, JS, ... l STATIC_URL # Folder for static files l TEMPLATE_DIRS
# Template folder

References

- Adrian Holovaty, Jacob Kaplan-Moss, “The Definitive Guide to Django: Web Development Done Right,” Apress, 2008.
- Django Software Foundation, “Django Documentation”, Release ??.
- Documentation (https://docs.djangoproject.com)

Django Platform

Creating a Project



Pure Python
Django
FastAPI
Flask
Google App Engine
Pyramid
Scientific
Angular CLI
Bootstrap
Express
HTML5 Boilerplate
Next.js
Node.js
React
React Native
Vite
Vue.js

Location: /Users/helder/Documents/Django/webproj

Python Interpreter: Python 3.11

New environment using Virtualenv

Location: /Users/helder/Documents/Django/webproj/venv

Base interpreter: Python 3.11 /usr/local/bin/python3.11

Inherit global site-packages

Make available to all projects

Previously configured interpreter

Interpreter: Python 3.11 /usr/local/bin/python3.11

Add Interpreter

More Settings

Template language: Django

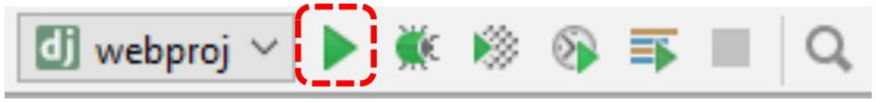
Templates folder: app/templates

Application name: app

☑ Enable Django admin

Web Application Execution

- Execution through PyCharm
  - Run/Run 'project' option (Shift + F10) • or icon



  - and click on the link http://127.0.0.1:8000

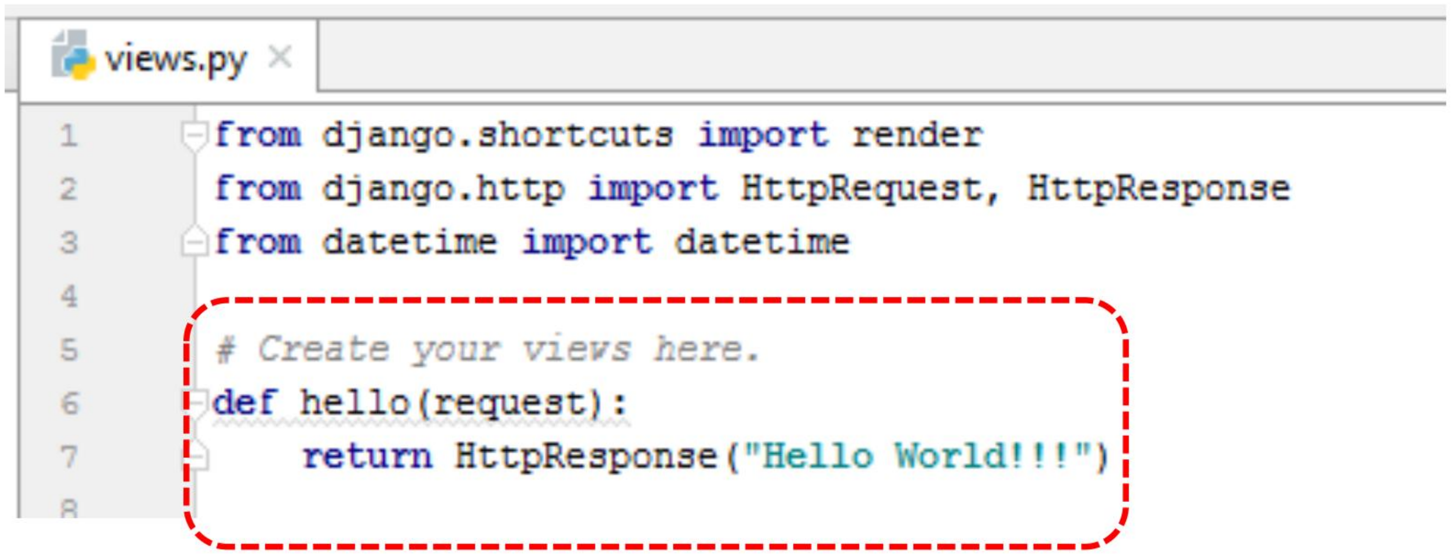
- Or execution in command line
  - inside the project folder, run: • “python manage.py runserver” • and open the browser with the URL: http://localhost:8000

Django Platform

Views

View - Creation

- In the “app/views.py” file, insert a view by defining a function
- Example:

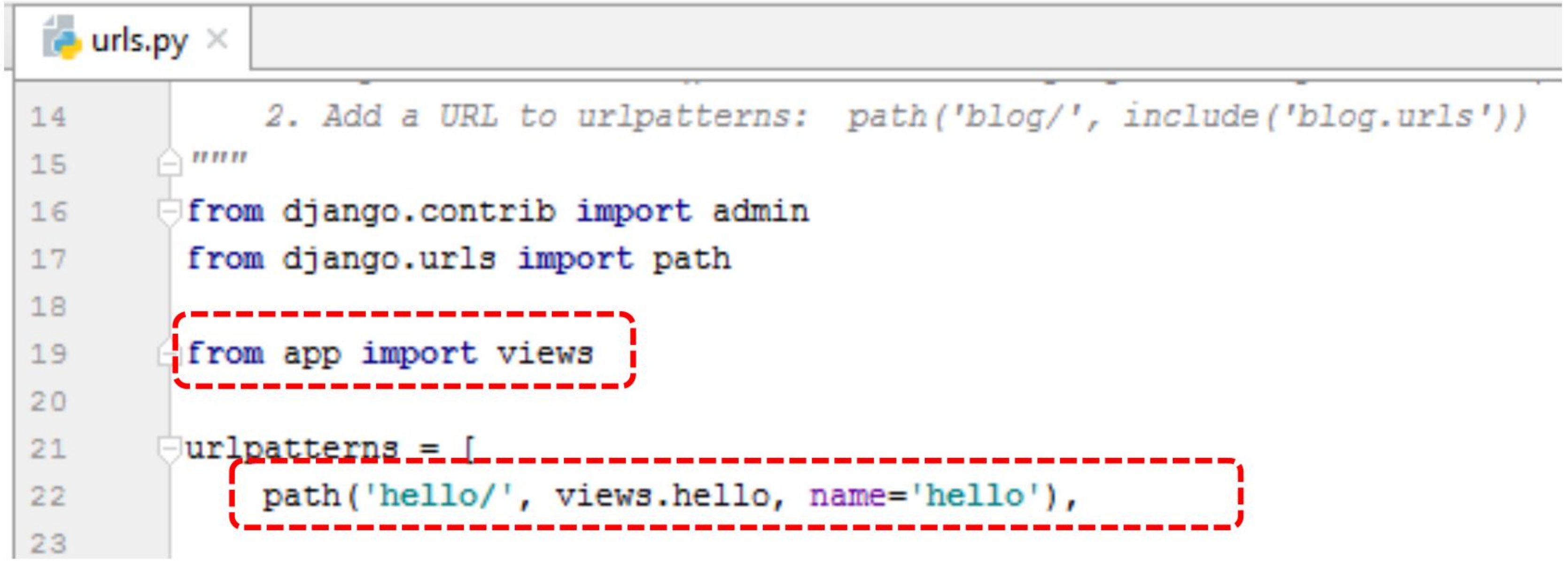


views.py

```python
from django.shortcuts import render
from django.http import HttpRequest, HttpResponse
from datetime import datetime

# Create your views here.
def hello(request):
    return HttpResponse("Hello World!!!")

```

URL configuration

- In the “Project_Name/urls.py” file, insert a *route* to the *view*

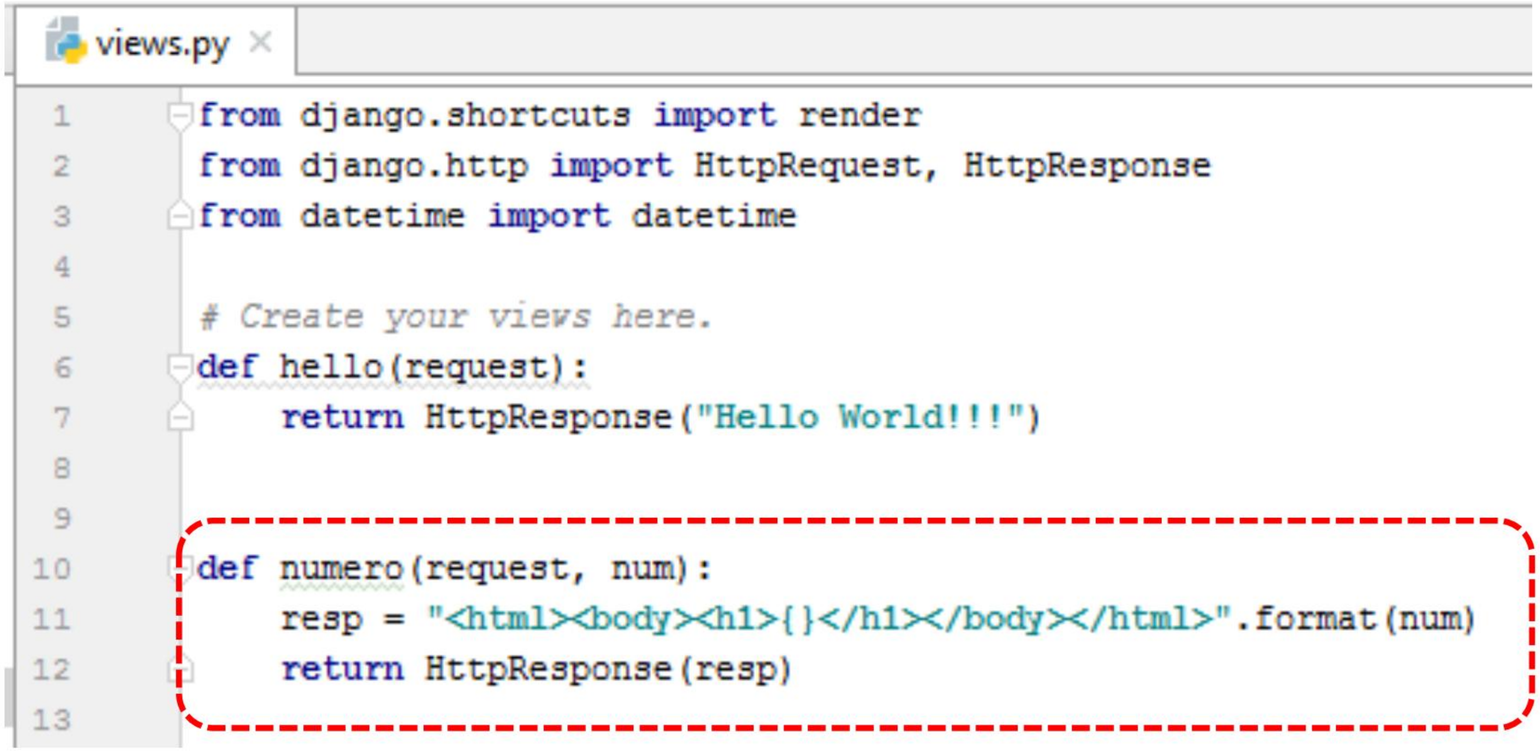


urls.py

```python
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path

from app import views

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

```

View - New

- In the “app/views.py” file, insert another *view function:*

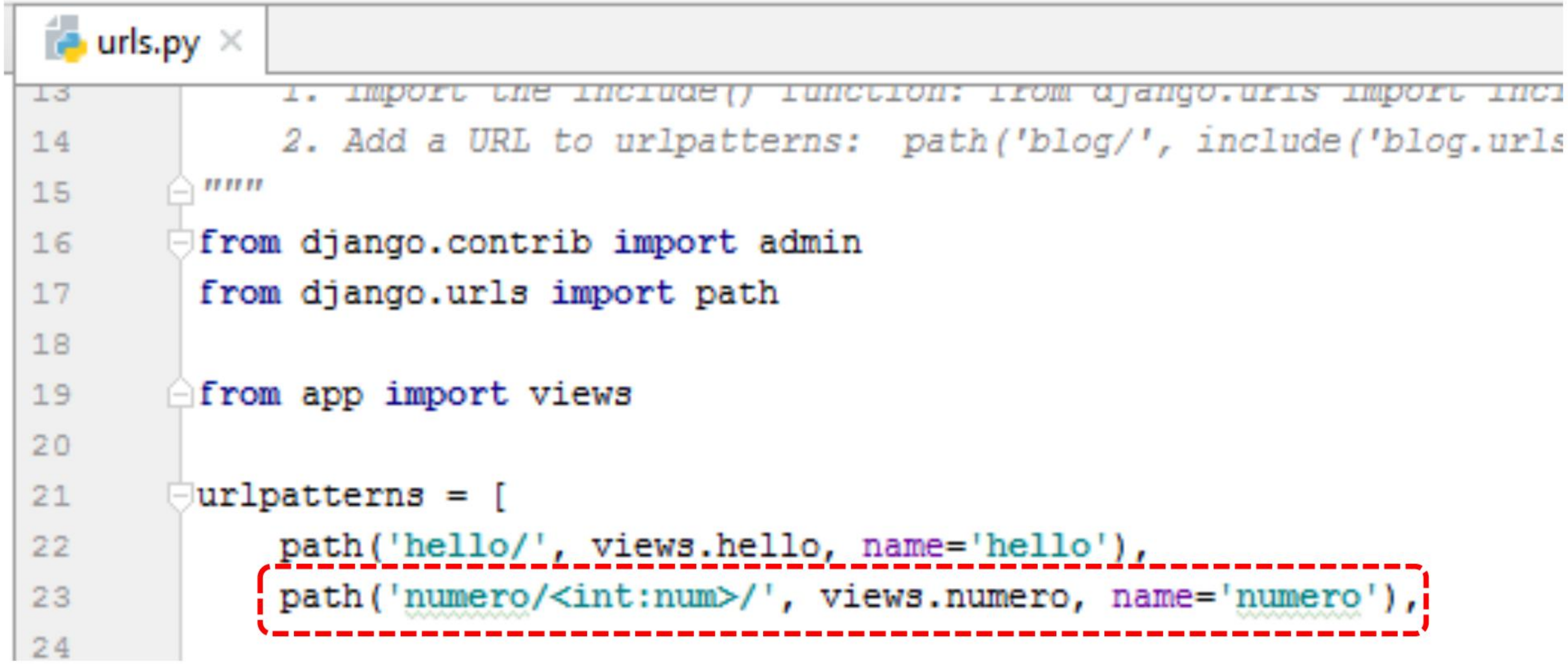


views.py

```
from django.shortcuts import render
from django.http import HttpRequest, HttpResponse
from datetime import datetime

# Create your views here.
def hello(request):
    return HttpResponse("Hello World!!!")


def numero(request, num):
    resp = "<html><body><h1>{}</h1></body></html>".format(num)
    return HttpResponse(resp)
```

Configuring the new URL

- In the “Project_Name/urls.py” file, insert another *route* for the *view*



urls.py

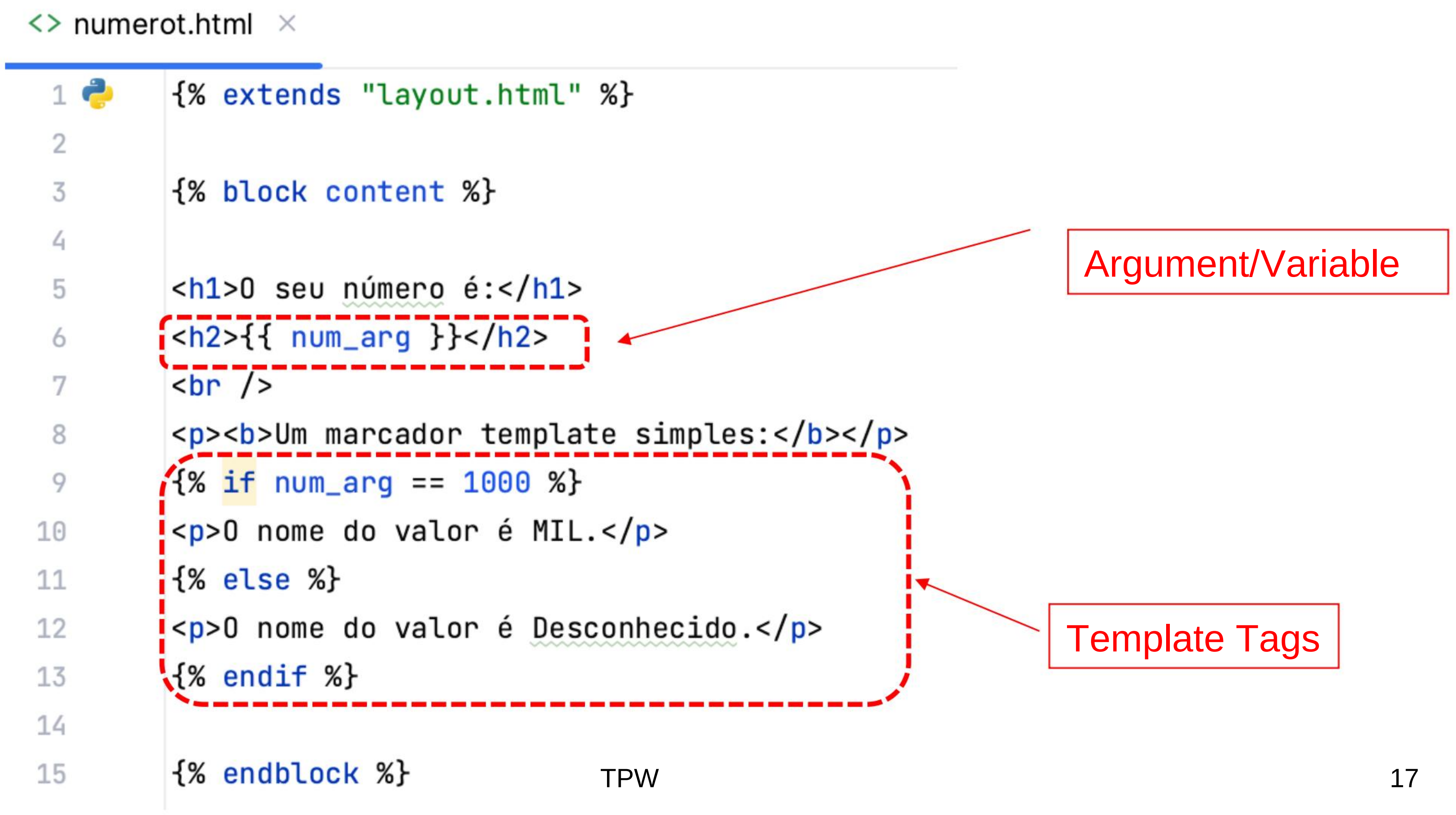
```
    1. Import the include() function: from django.urls import incl
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls
"""
from django.contrib import admin
from django.urls import path

from app import views

urlpatterns = [
    path('hello/', views.hello, name='hello'),
    path('numero/<int:num>/', views.numero, name='numero'),
```

Django Platform

Templates

Template - Creation

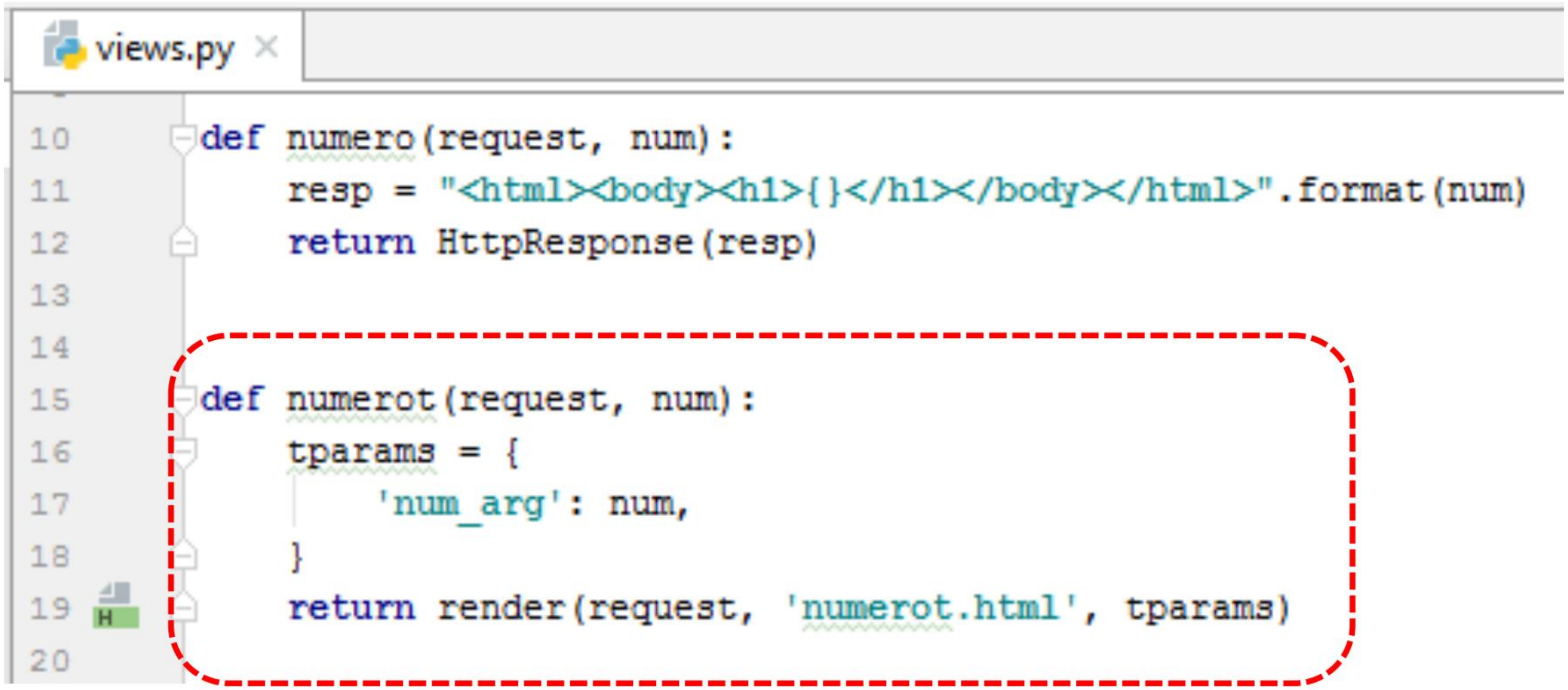
- In the "templates" folder, create the "numerot.html" file



```
{% extends "layout.html" %}

{% block content %}

<h1>O seu número é:</h1>
<h2>{{ num_arg }}</h2>
<br />
<p><b>Um marcador template simples:</b></p>
{% if num_arg == 1000 %}
<p>O nome do valor é MIL.</p>
{% else %}
<p>O nome do valor é Desconhecido.</p>
{% endif %}

{% endblock %}
```

Template – New *View*

- In the “app/views.py” file, insert another *view function:*

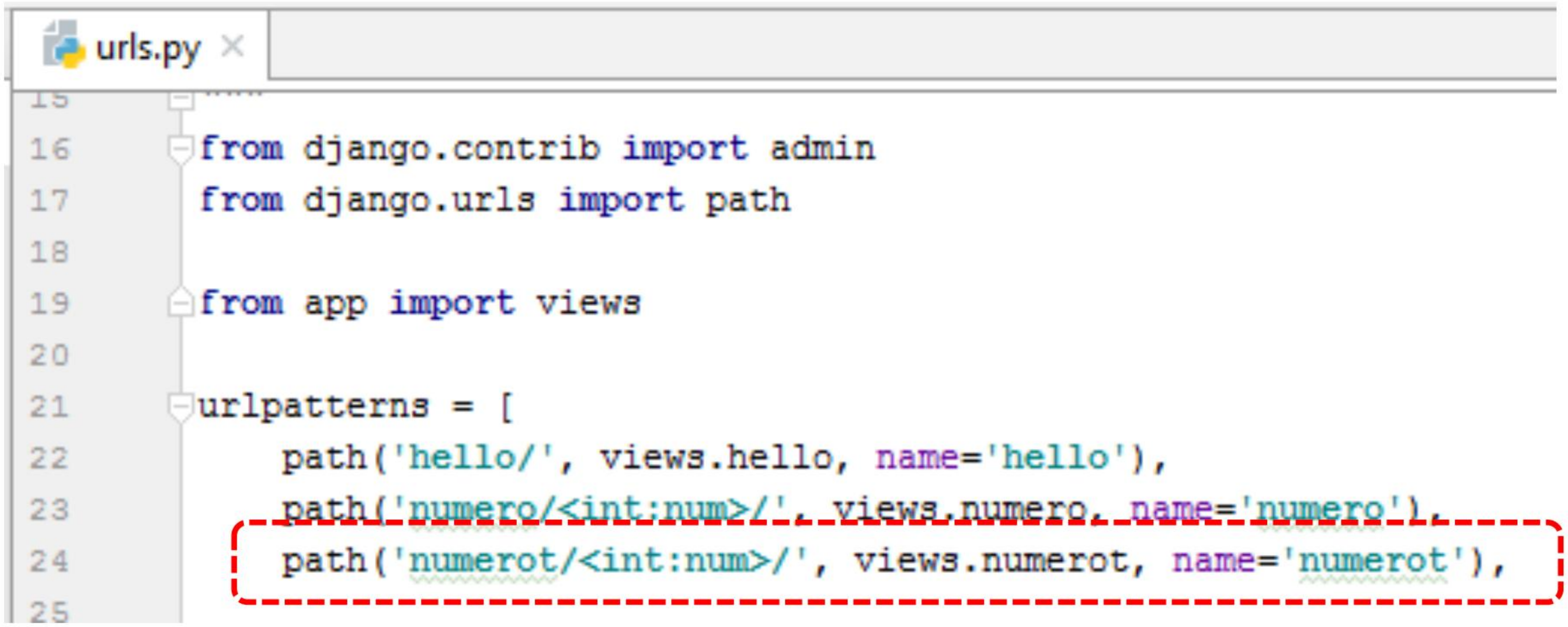


views.py

```
def numero(request, num):
    resp = "<html><body><h1>{}</h1></body></html>".format(num)
    return HttpResponse(resp)


def numerot(request, num):
    tparams = {
        'num_arg': num,
    }
    return render(request, 'numerot.html', tparams)
```

Configuring the new URL

- In the “Project_Name/urls.py” file, insert another *route* for the *view*



urls.py

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

from app import views

urlpatterns = [
    path('hello/', views.hello, name='hello'),
    path('numero/<int:num>/', views.numero, name='numero'),
    path('numerot/<int:num>/', views.numerot, name='numerot'),
```

Django Platform

static files

static files

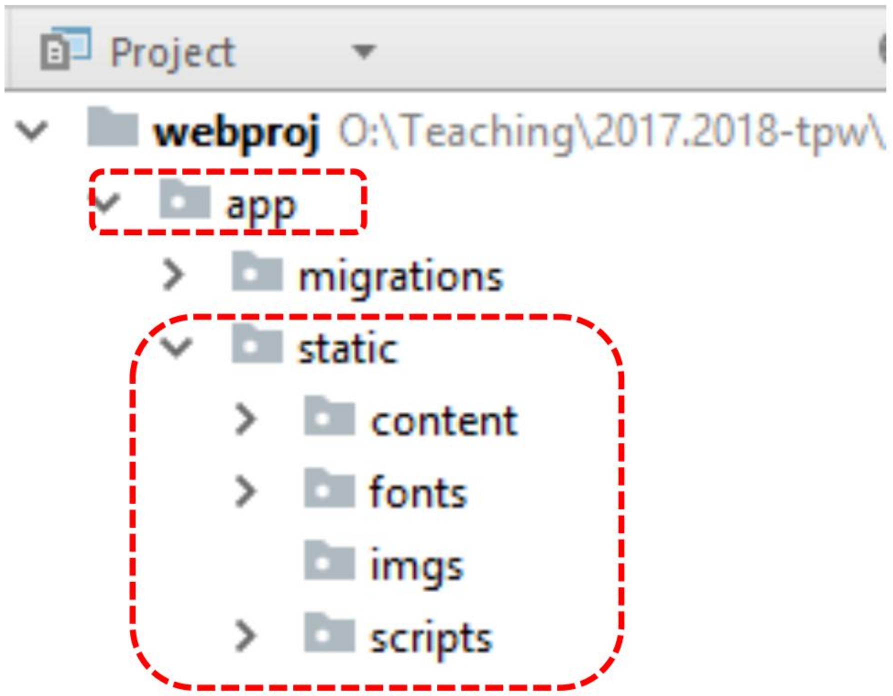
- *Static files* are files that are simply intended to be referenced and served to the client, without any prior processing.
- Their access is public, as the client only needs to have the URL for them.
- Examples:
  - Images (jpg, png, etc.)
  - Style Sheets (CSS)
  - Scripts (JavaScript)

Static Files - Location

- Files called *static files* reside in pre-determined folders, inside or outside the “app”.

- Example:
  - Inside the “app/ static”

  folder • There are the “content”, “fonts”,
  “scripts” folders • Other folders can be added, such as the “imgs” folder


Project

webproj O:\Teaching\2017.2018-tpw\

app

migrations

static

content

fonts

imgs

scripts

Static Files - Configuration

• In the “settings.py” file:

• the 'django.contrib.staticfiles' module must appear in the installed applications “INSTALLED_APPS” • the URL

prefix for *static files* must be defined:

• STATIC_URL = '/static/' •

the *static files* folder must be defined:

• STATIC_ROOT = os.path.join(BASE_DIR, 'app/static')

• Documentation: • In

debug and production mode •

https://docs.djangoproject.com/en/4.0/howto/static-files/ • https://docs.djangoproject.com/en/4.0/howto/static-files/deploym

Static Files - Usage

- Resource reference, mode 1: • This mode uses absolute URLs <link

```
rel="stylesheet" href="static/content/style.css" />
<script src="static/scripts/jquery-1.10.2.min .js"></
script> <script src="static/scripts/main.js"></script>
```

- Resource reference, mode 2: • This mode uses relative URLs {% load static %}

```
<link
rel="stylesheet" href="{% static "content/style.css" %}" />
<script src=" {% static "scripts/jquery-1.10.2.min.js" %}"></
script> <script src="{% static "scripts/main.js" %}"></script>
```