

Tecnologias e Programação Web

Introdução à Plataforma Django

Introdução



- Django é uma plataforma gratuita e open source, escrita em Python, para o desenvolvimento de aplicações web.
- Tomou o nome de um famoso guitarrista “Django Reinhardt”
- É mantida pela Django Software Foundation (DSF), uma organização independente
- Fomenta o desenvolvimento rápido, limpo e pragmático
- Criada em 2003, tornou-se open source em 2005

Caraterísticas

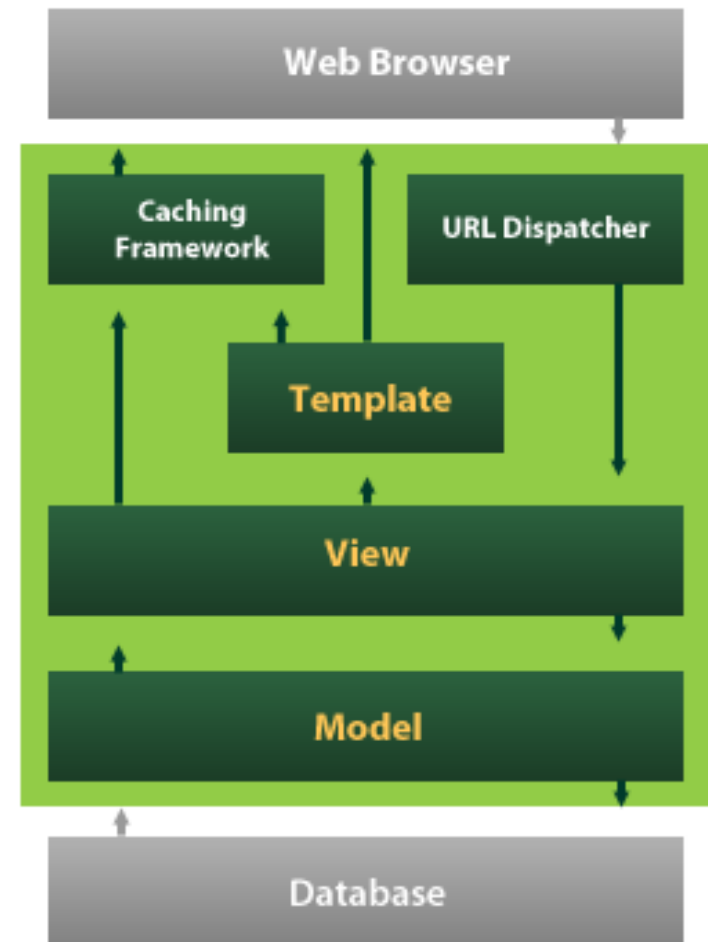


- Segue, parcialmente, o padrão MVC
- Possui um ORM (*Object Relational Mapper*) para processar dados
- Focada na automatização, aderindo ao princípio DRY (Don't Repeat Yourself)
- Usa um sistema de templates
- Sistema de personalização Admin, para facilitar o CRUD
- Desenho elegante de routing de URLs
- Possui um light web server embutido (para testes)
- Possibilita a utilização de middleware personalizado
- Possui facilidades para: autenticação, internacionalização e caching

Arquitetura



Models	Descreve os dados
Views	Controla o que os utilizadores vêem
Templates	Controlo como o vêem
URL Dispatcher	Expedidor de URLs



Estrutura do Projeto Django



```
webproj/ ----- Pasta para o projeto. Pode ter qualquer nome.  
manage.py -- Utilitário em commando de linha para interagir com o projeto.  
webproj/ --- Pacote do projeto. Nome usado para imports.  
    __init__.py --- Ficheiro que define esta pasta como um pacote, em Python.  
    settings.py --- Configurações do projeto Django.  
    urls.py ----- Mapping/routing das URLs para este projeto.  
    wsgi.py ----- Um ponto de entrada para webservers compatíveis com WSGI.  
app/ ----- Aplicação web individual, podendo coexistir várias.  
    templates/ ---- Ficheiros HTML, invocados pelas views.  
    static/ ----- CSS, JS, imagens, etc. - configurável em "settings.py"  
    __init__.py  -- Ficheiro que define esta pasta como um pacote, em Python.  
    views.py ----- Recebe os pedidos dos clientes e devolve as respostas.  
    models.py ---- Modelos dos dados.  
    admin.py ----- Criação automática de interface para o modelo de dados.  
    forms.py ----- Permite a receção de dados enviados pelos clients.
```

Settings



- O ficheiro settings.py do projeto Django sobrepõe-se ao ficheiro `<python>/Lib/sitepackages/django/conf/global_settings.py`
- Atributos:
 - DEBUG # True ou False
 - DATABASES ENGINE # 'mysql', 'sqlite3', 'oracle' ... etc.
 - ROOT_URLCONF # Configuração de routing das URLs
 - MEDIA_ROOT # Para ficheiros enviados pelo utilizador (user-uploaded)
 - MEDIA_URL # Para ficheiros multimedia
 - STATIC_ROOT # Pasta para ficheiros estáticos como CSS, JS, ...
 - STATIC_URL # Pasta de ficheiros estáticos
 - TEMPLATE_DIRS # Pasta de templates

Referências



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



Plataforma Django

Criação de Um Projeto



- 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:

Python Interpreter: Python 3.11

☐ New environment using

Location:

Base interpreter:

☐ Inherit global site-packages

☐ Make available to all projects

☒ Previously configured interpreter

Interpreter: [Add Interpreter](#)

More Settings

Template language:


Templates folder:

Application name:

☒ Enable Django admin

Execução da Aplicação Web



- Execução através do PyCharm
 - opção Run/Run 'project' (Shift + F10)
 - ou no icon The image shows a snippet of the PyCharm interface. It features a dropdown menu with 'dj webproj' selected, followed by a green play button icon (Run) which is highlighted with a red dashed border. To the right of the play button are several other icons: a green bug (Debug), a green play button with a checkmark (Run with Coverage), a green play button with a magnifying glass (Run with Profiler), a green play button with a list icon (Run with Debugger), a grey square (Run with External Tools), and a magnifying glass (Search).
 - e clicar no link `http://127.0.0.1:8000`
- Ou execução em comando linha
 - dentro da pasta do projeto executar:
 - “python manage.py runserver”
 - e abrir o browser com a URL: <http://localhost:8000>



Plataforma Django

Views

View - Criação



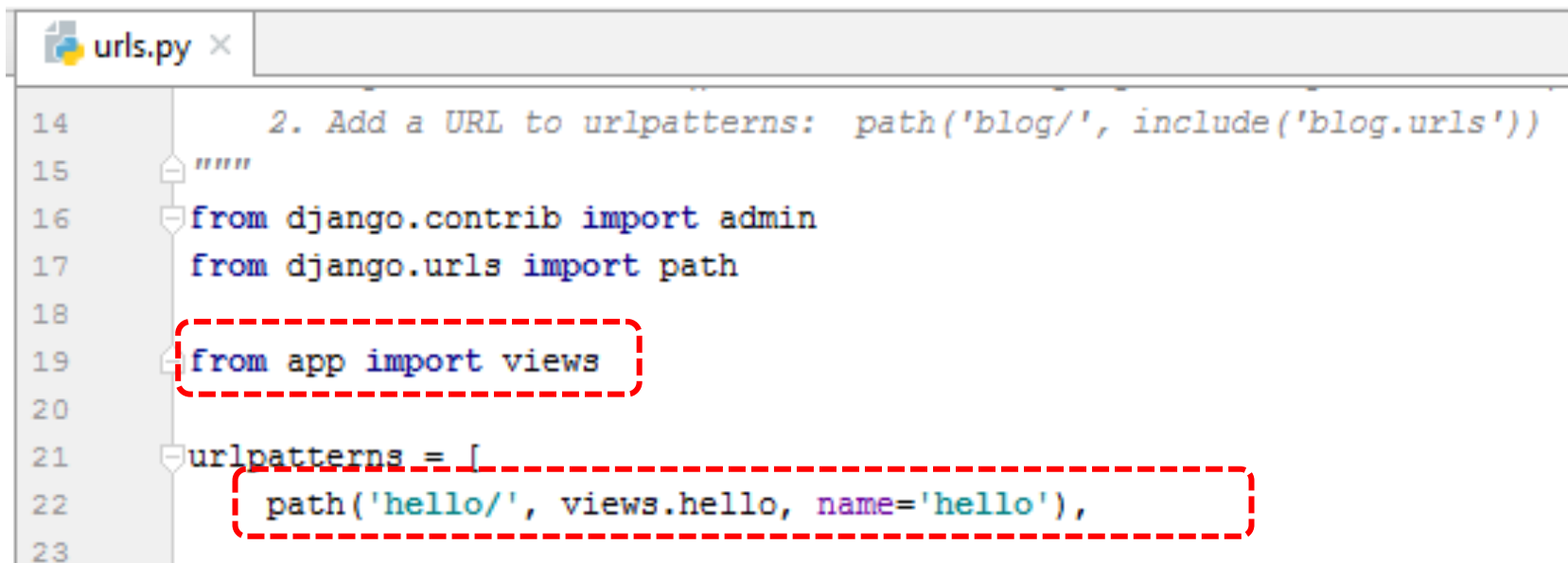
- No ficheiro “app/views.py” inserir uma view através da definição de uma função
- Exemplo:

```
views.py x
1 from django.shortcuts import render
2 from django.http import HttpRequest, HttpResponse
3 from datetime import datetime
4
5 # Create your views here.
6 def hello(request):
7     return HttpResponse("Hello World!!!")
8
```

Configuração da URL



- No ficheiro “Nome_Projeto/urls.py” inserir uma *route* para a *view*



```
14      2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15      """
16      from django.contrib import admin
17      from django.urls import path
18
19      from app import views
20
21      urlpatterns = [
22          path('hello/', views.hello, name='hello'),
23
```

View - Nova



- No ficheiro “app/views.py” inserir mais uma *view function*:

```
views.py x
1 from django.shortcuts import render
2 from django.http import HttpRequest, HttpResponse
3 from datetime import datetime
4
5 # Create your views here.
6 def hello(request):
7     return HttpResponse("Hello World!!!")
8
9
10 def numero(request, num):
11     resp = "<html><body><h1>{}</h1></body></html>".format(num)
12     return HttpResponse(resp)
13
```

Configuração da nova URL



- No ficheiro “Nome_Projeto/urls.py” inserir mais uma *route* para a *view*

```
13 1. import the include() function: from django.urls import incl
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls
15 """
16 from django.contrib import admin
17 from django.urls import path
18
19 from app import views
20
21 urlpatterns = [
22     path('hello/', views.hello, name='hello'),
23     path('numero/<int:num>/', views.numero, name='numero'),
24 ]
```



Plataforma Django

Templates

Template - Criação



- Na pasta “templates”, criar o ficheiro “numerot.html”

<> numerot.html x

```
1 {% extends "layout.html" %}
2
3 {% block content %}
4
5 <h1>0 seu número é:</h1>
6 <h2>{{ num_arg }}</h2>
7 <br />
8 <p><b>Um marcador template simples:</b></p>
9 {% if num_arg == 1000 %}
10 <p>0 nome do valor é MIL.</p>
11 {% else %}
12 <p>0 nome do valor é Desconhecido.</p>
13 {% endif %}
14
15 {% endblock %}
```

Argumento/Variável

Template Tags

Template – Nova View



- No ficheiro “app/views.py” inserir mais uma *view function*:

```
views.py x
10 def numero(request, num):
11     resp = "<html><body><h1>{}</h1></body></html>".format(num)
12     return HttpResponse(resp)
13
14
15 def numerot(request, num):
16     tparams = {
17         'num_arg': num,
18     }
19     return render(request, 'numerot.html', tparams)
20
```

Configuração da nova URL



- No ficheiro “Nome_Projeto/urls.py” inserir mais uma *route* para a *view*



```
15
16 from django.contrib import admin
17 from django.urls import path
18
19 from app import views
20
21 urlpatterns = [
22     path('hello/', views.hello, name='hello'),
23     path('numero/<int:num>/', views.numero, name='numero'),
24     path('numerot/<int:num>/', views.numerot, name='numerot'),
25 ]
```



Plataforma Django

Static Files

Static Files

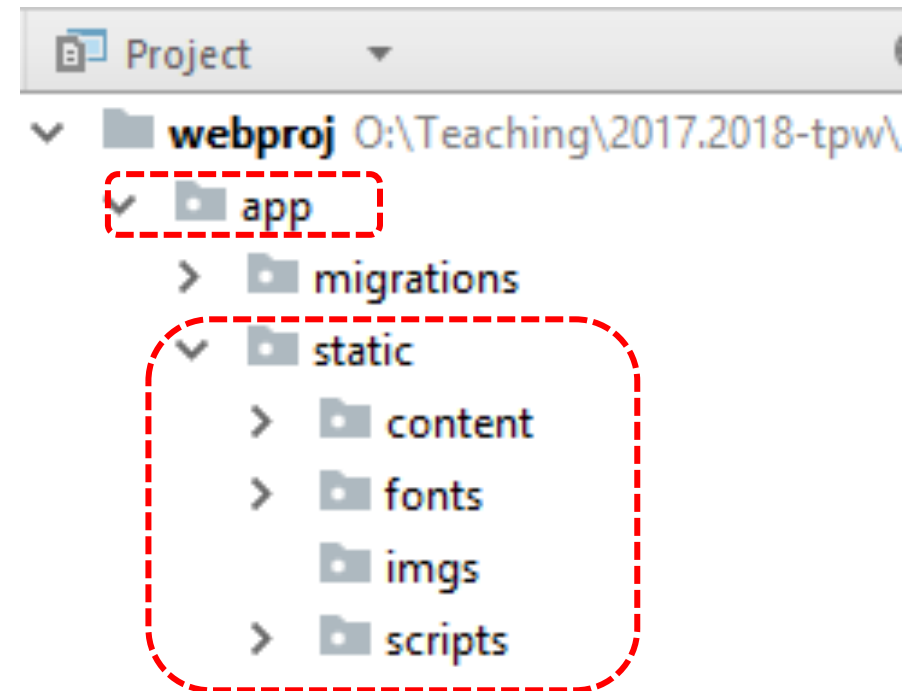


- *As static files* são ficheiros que se pretende simplesmente referenciar e servir ao cliente, sem qualquer processamento prévio.
- O seu acesso é público, pois o cliente apenas necessita de possuir a URL para os mesmos.
- Exemplos:
 - Imagens (jpg, png, etc.)
 - Style Sheets (CSS)
 - Scripts (JavaScript)

Static Files - Localização



- Os ficheiros denominados por *static files* residem em pastas pré-determinadas, dentro ou fora da “app”.
- Exemplo:
 - Dentro da pasta “app/static”
 - Encontram-se as pastas “content”, “fonts”, “scripts”
 - Podem ser adicionada ainda outras pastas como a pasta “imgs”



Static Files - Configuração



- No ficheiro “settings.py”:
 - o módulo `'django.contrib.staticfiles'` deve aparecer nas aplicações instaladas “INSTALLED_APPS”
 - o prefixo da URL para *static files* deve estar definido:
 - `STATIC_URL = '/static/'`
 - a pasta das *static files* deve estar definido:
 - `STATIC_ROOT = os.path.join(BASE_DIR, 'app/static')`
- Documentação:
 - Em modo debug e produção
 - <https://docs.djangoproject.com/en/4.0/howto/static-files/>
 - <https://docs.djangoproject.com/en/4.0/howto/static-files/deployment/>

Static Files - Uso



- Referência de recursos, modo 1:

- Este modo usa URLs absolutos

```
<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>
```

- Referência de recursos, modo 2:

- Este modo usa URLs relativos

```
{% 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>
```