Tecnologias e Programação Web

Resumos
2015/2016

João Alegria | 68661

8º Capítulo

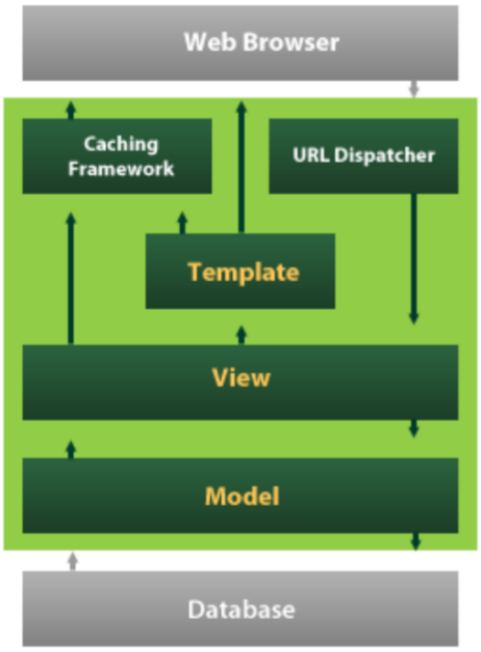
Introdução ao Django

• Django é uma plataforma gratuita e open source, escrita em Python, para o desenvolvimento de aplicações web.
• É mantida pela Django Software Foundation (DSF), uma organização independente
• Fomenta o desenvolvimento rápido, limpo e pragmático
• Versão atual: 1.8

Características

• Arquitetura MVC (MVT, na verdade)
• 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
  - caching

Arquitetura



Models - Descreve os dados

Views - Controla o que os utilizadores vêm

Templates - Controlo como o vêm

Controller - 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

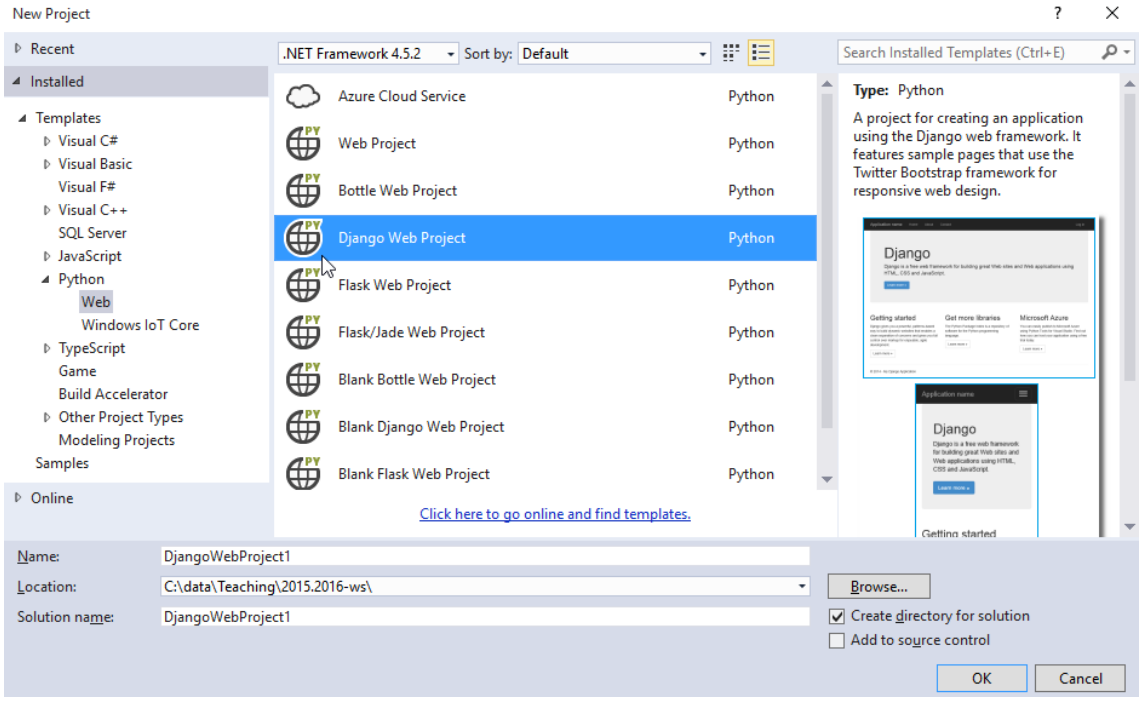
- ***Atributos:***

- 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

Documentação

https://docs.djangoproject.com

Criação de um Projeto



Execução da Aplicação Web

- Execução através do Visual Studio
  - opção Debug/Start Debugging (F5)

OU

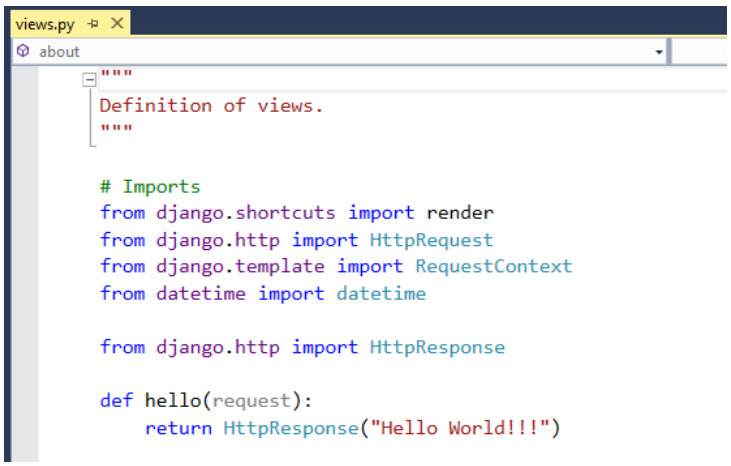
- Execução em linha de comandos
  - Dentro da pasta do projeto executar:
    - `python manage.py runserver`
    - abrir o browser com a URL: http://localhost:8000

Plataforma Django

View

- ***Criação:***

- No ficheiro "app/views.py" inserir uma *view* através da definição de uma função.



• ***Configuração da URL:***

- No ficheiro "*NomeProjeto*/urls.py" inserir uma *route* para a *view*.

urls.py

```
from datetime import datetime
from django.conf.urls import patterns, url
from app.forms import BootstrapAuthenticationForm

from app import views

# Uncomment the next lines to enable the admin:
# from django.conf.urls import include
# from django.contrib import admin
# admin.autodiscover()

urlpatterns = patterns('',
    # Examples:
    url(r'^hello$', 'app.views.hello', name='hello'),
```

• ***Nova:***

- No ficheiro "*app*/views.py" inserir mais uma *view function*.

views.py — about

```
# Imports
from django.shortcuts import render
from django.http import HttpRequest
from django.template import RequestContext
from datetime import datetime

from django.http import HttpResponse

def hello(request):
    return HttpResponse("Hello World!!!")

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

• ***Configuração da nova URL:***

- No ficheiro "*NomeProjeto*/urls.py" inserir mais uma *route* para a *view*.

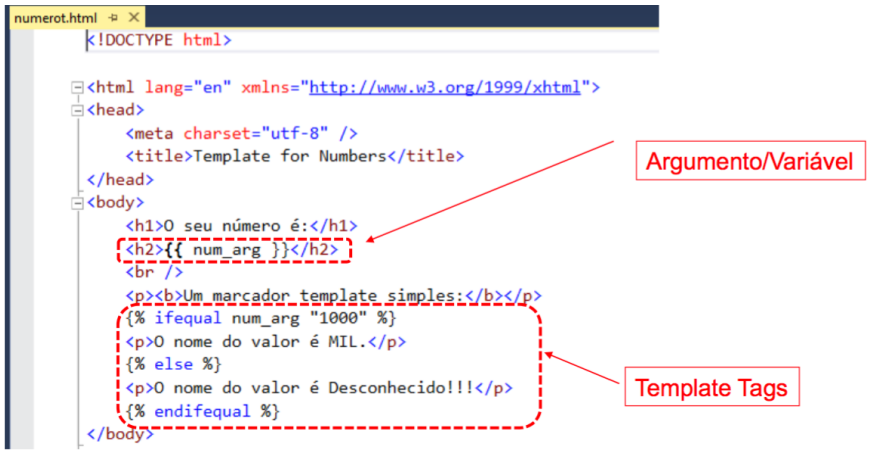
urls.py

```
from datetime import datetime
from django.conf.urls import patterns, url
from app.forms import BootstrapAuthenticationForm

from app import views

# Uncomment the next lines to enable the admin:
# from django.conf.urls import include
# from django.contrib import admin
# admin.autodiscover()

urlpatterns = patterns('',
    # Examples:
    url(r'^hello$', 'app.views.hello', name='hello'),

    url(r'^numero/(\d+)$', 'app.views.numero', name='numero'),
```

Templates

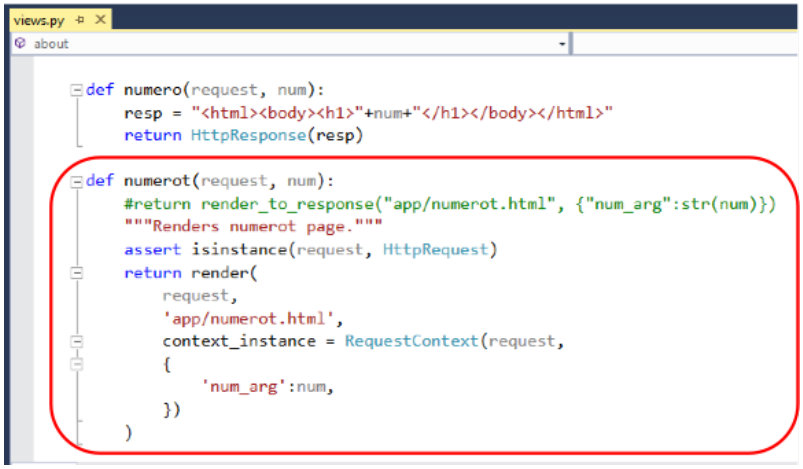
- ***Criação:***

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



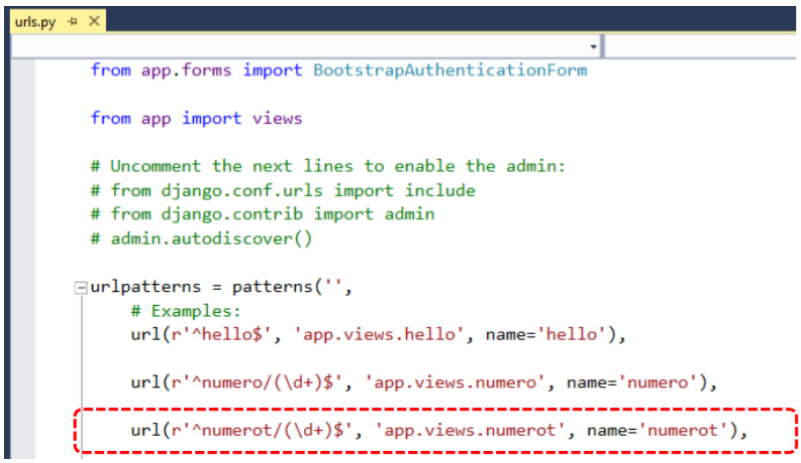
- ***Nova View:***

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



- ***Configuração da nova URL:***

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



Static Files

• As *static files* são ficheiros que se pretende simplesmente referenciar e servir ao cliente, sem qualquer processamento prévio.

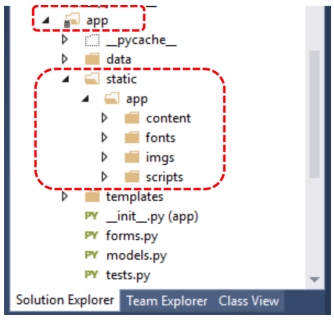
• O seu acesso é público, pois o cliente apenas necessita de possuir a URL para os mesmos.

- Exemplos:
  - Imagens (jpg, png, etc.)
  - Style Sheets (CSS)
  - Scripts (JavaScript)

• ***Localização:***

- Os ficheiros denominados por *static files* residem em pastas pré-determinadas, dentro ou fora da "app".

- Exemplos:
  - Dentro da pasta "app/static/app"
    - Encontram-se as pastas "content", "fonts", "scripts"
    - Foi ainda adicionada a pasta "imgs"



• ***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 configurado, exemplo:

    STATIC_URL = '/static/'
  - Documentação:
    https://docs.djangoproject.com/en/1.8/howto/static-files/
    https://docs.djangoproject.com/en/1.8/howto/static-files/deployment/

• ***Uso:***

Referência de recursos - Modo 1 URLs absolutos	**Referência de recursos - Modo 2** URLs relativos

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

AJAX

• O uso do Ajax em combinação com um framework, como o Django, deve ser um processo ponderado. Devem ser observadas as seguintes regras:
  - Necessidade de carregar dados para uma página já carregada no browser.
  - Necessidade de alterar uma parte de uma página já carregada no browser.
• O seu uso processa-se da mesma forma como os *request* normais
• Para pedido de recursos estáticos (*static files*), devem ser usadas URLs como os descritos para este tipo de ficheiros
• Para outros pedidos, deve ser usado o método de acesso a *views*

• ***Layout:***
- Necessidade de alteração de uma parte da página já carregada no browser.
  - No script:

```
var url = "layouts #lay01";
$(container).load(url, function (result, status, shr)
{…})
```

  - Na view:

```
def layouts(request):
 return render(request, 'app/layouts.html',)
```

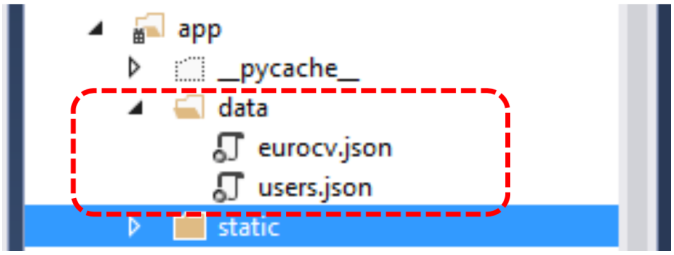
• ***Dados:***
- Necessidade de alteração de uma parte da página já carregada no browser.
  - No script:

```
var file = "eurocv";
var strURL = "datajson/" + file;
$.ajax({ url: strURL, type: "GET", … });
```

  - Na view:

```
def datajson(request, fname):
 filename = path.join(path.dirname(__file__), "data",
fname + ".json")
 filej = open(filename)
 resp = json.load(filej)
 filej.close()
 return JsonResponse(resp)
```

• ***Dados (local):***
- Os ficheiros de dados devem localizar-se numa pasta para esse efeito e não devem ser colocados juntamente com as *static files*.
- Exemplo:
  - Dentro da pasta "app", criar uma pasta específica "data", como sugerido na figura



• ***URLs:***

- Para o funcionamento dos pedidos Ajax, usando *views*, é necessário definir o *routing* das urls no ficheiro “urls.py”.
- Exemplo:

```
url (r'^layouts', 'app.views.layouts', name = 'layouts'),
url (r'^datajson/(.+)', 'app.views.datajson', name = 'datajson'),
```