Django

Introdução:
Instalação, config. inicial de um projeto

Ely – elydasilvamiranda [at] gmail.com

Django

- Framework de alto nível feito em Python
- Framework Full Stack:
  - desenvolve-se todos os aspectos de uma aplicação somente com ele.
- Criado em 2003 e disponibilizado como Opensource;
- Ágil e plugável: foco na automação e nos princípios DRY (Don't Repeat Yourself);

Principais elementos

- ORM: Mapeamento Objeto-Relacional;
- Interface administrativa;
- Internacionalização;
- Sistema de templates;
- Infraestrutura de Cache;
- Validação de formulários;
- Gerenciador de perfis, autorização e autenticação;

Instalando o Django

- Para instalar o Django usaremos o pip;
- O pip é a ferramenta de instalação de pacotes do Python;
- Baixa-se diretamente da web pacotes com as mais diversas funcionalidades;
- Sintaxe:

  pip install nome_do_pacote==versão
- Instalando o Django:

  >>> pip install django==1.11

Verificando a versão

- Verificando a versão instalada:
  >>> import django

  >>> django.get_version()

  ou:

  >>> python -m django --version

5

Criando um projeto

- Em Django devemos criar projetos;
- Projetos podem ter mais de uma aplicação;
- Para nosso estudo, criaremos:
  - Um projeto chamado **connectedin**;
  - Uma aplicação inicial chamada **perfis**;
- Deve-se utilizar o comando django-admin.py para criar projetos;

  Nota: esse arquivo deve estar no path do S.O.

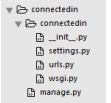
6

Criando um projeto

- Sintaxe:

  django-admin.py startproject <nome_projeto>
- Os nomes de projetos e aplicações não devem usar palavras reservadas da linguagem ou nomes específicos do django;
- Criando um projeto chamado connectedin:

  >>> django-admin.py startproject connectedin
- Ao criar um projeto, cria-se uma pasta com o nome do projeto:



7

Criando um projeto

- Sobre as pastas e arquivos criados:
- connectedin: pasta onde o projeto está guardado;
  - connectedin: projeto em si que não deve ser renomeada;
    - __init__.py: arquivo vazio (indica um package);
    - settings.py: arquivo de configuração do projeto;
    - urls.py: definições de URLs do projeto;
    - wsgi.py: protocolo parecido com fastCGI serve HTTP;
  - manage.py: utilitário semelhante ao django-admin.py.

8

Criando o banco

- O manage.py é um script para gerenciar a aplicação com Django;
- Deve ser executado através do interpretador python dentro da pasta do projeto;
- O próximo passo é configurar o BD do projeto;
- Por padrão, o Django vem com o SQLite;
- Na pasta do projeto digite:

  >>> python manage.py migrate

- O arquivo connectedin/db.sqlite3 então representa o arquivo do banco.

9

Configs. de banco

- O tipo de banco são configuradas no arquivo settings.py:

```
# trecho do arquivo conf.py
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

- Backends suportados por padrão:
  - 'django.db.backends.sqlite3','django.db.backends.postgresql', 'django.db.backends.mysql' e'django.db.backends.oracle'

10

Configs. de banco

- Backends disponibilizados por terceiros:
  - https://docs.djangoproject.com/en/1.11/ref/databases/#third-party-notes
- Configurando usuários e senhas:
  - https://docs.djangoproject.com/en/1.11/ref/settings/#std:setting-DATABASES

11

Ajuda do manage.py

- Para obter ajuda de algum comando do manage.py, utilize:

  manage.py help <comando>

  Ex:

  manage.py help migrate

12

Testando o projeto

- O Django possui um servidor web interno para ser usado no ambiente de desenvolvimento;

  Nota: esse é um servidor de testes, não devendo ser usado em "produção"

- O servidor local possui:
  - recarga automática de módulos;
  - serve os arquivos estáticos (javascripts, css, imagens, etc.) sem configurações adicionais;
- O comando para executa-lo é:
  >>> python manage.py runserver porta.

  Onde a porta é opcional, por padrão é a 8000

Testando o projeto

- Usa-se o comando runserver para "subir" o nosso projeto:

  >>> python manage.py runserver

```
D:\temp\connectedin>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
March 21, 2017 - 21:01:02
Django version 1.7.4, using settings 'connectedin.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[21/Mar/2017 21:01:14] "GET / HTTP/1.1" 200 1759
[21/Mar/2017 21:01:14] "GET /favicon.ico HTTP/1.1" 404 1932
```

- Após isso, deve-se testar a aplicação pelo navegador: http://localhost:8000

Criando uma aplicação

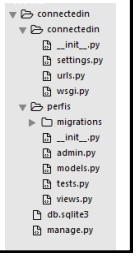
- Um projeto pode ter várias aplicações;
- Uma aplicação no Django:
  - é uma forma de dividir a responsabilidade dentro do projeto;
  - é um módulo que confina dentro dele determinada responsabilidade;
- As aplicações ficam armazenadas dentro da pasta do projeto.

Criando uma aplicação

- Para criar uma nova aplicação, deve-se entrar no diretório do projeto pelo prompt de comando;
- Aplicações são criadas através do comando:

  python manage.py startapp <nome_aplicação>
- Criando uma aplicação chamada perfis:

  >>> python manage.py startapp perfis



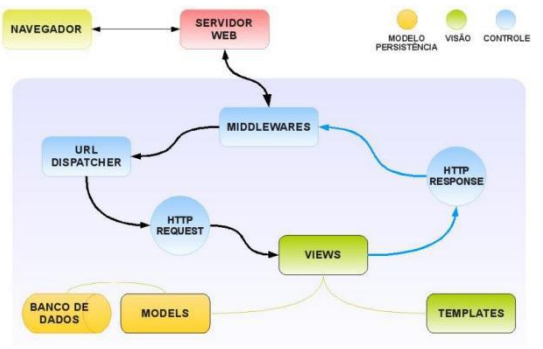
Registrando uma aplicação

- Deve-se registrar a aplicação no arquivo settings.py do projeto;
- Nesse arquivo, há uma declaração chamada INSTALLED_APPS;
- Cada nova aplicação deve ser adicionada como o último elemento usando aspas simples:

```
# código anterior omitido
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'perfis'
)
# código posterior omitido
```

17

Diagrama de requisições



18

Views

- Views é um módulo python que agrupa um conjunto de actions;
- Toda view deve:
  - receber um objeto “HTTPRequest” como primeiro parâmetro;
  - retornar um objeto “HTTPResponse” como resposta.
- O objeto “HTTPRequest” é fornecido automaticamente pelo Django;
- O objeto “HTTPResponse” é de responsabilidade do desenvolvedor.

19

Criando uma view

- Dentro de cada aplicação do projeto existe um arquivo chamado views.py;
- Nesse arquivo definimos a "resposta" para o usuário após uma requisição via browser;
- Para definir uma view, deve-se editar esse arquivo:

```
# connectedin/perfis/views.py

from django.shortcuts import render
from django.http import HttpResponse

def index(request):
    return HttpResponse('Bem-vindo ao Connectedin')
```

20

Criando uma view

- Sobre o código anterior:
  - A função index (poderia ser qualquer nome) representa uma página a ser acessada;
  - Recebe um parâmetro chamado request, que representa a requisição do usuário;

    (https://docs.djangoproject.com/en/1.11/ref/request-response/#django.http.HttpRequest)
  - O objeto responsável pela resposta deve ser importado e instanciado;
  - A resposta contém um texto de boas vindas;
- Para acessar essa função, deve-se especificar uma rota através do sistema/módulo de rotas do Python;

Urls

- Urls é um módulo python responsável por realizar o roteamento de URLs do projeto;
- Esse módulo utiliza o mapeamento das URLs utilizando expressões regulares (regex);
- Todas as urls podem ficar em um unico arquivo urls.py;
- É recomendável que cada aplicação contenha seu próprio arquivo urls.py;
- Posteriormente o arquivo urls.py do projeto deve importar os módulos urls.py de cada aplicação.

Definindo uma URL

- As rotas do projeto são definidas no arquivo connectedin/connectedin/urls.py;
- Deve-se adicionar uma rota a mais ao final do arquivo:

```
# connectedin/connectedin/urls.py

from django.conf.urls import url
from django.contrib import admin
from perfis import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^$', views.index),
]
```


Definindo uma URL

- Argumentos da função URL:
  - O primeiro parâmetro é uma expressão regular que representa um caminho acessado via navegador;
  - O segundo é uma view que deve ser executada;
  - Existe um terceiro parâmetro chamado name, que será visto posteriormente;
- Mais sobre URLs:
  - https://docs.djangoproject.com/en/1.11/topics/http/urls/
- Expressões regulares:
  - https://docs.python.org/3/library/re.html#module-re

Testando a view

- Rode a aplicação:

  >>> python manage.py runserver

- Acesse o endereço:

  http://localhost:8000/

25

Prática

- Repita os passos abaixo e crie um projeto e uma aplicação:
  1. Crie um projeto chamado **mysite**;
  2. Crie o banco;
  3. Suba o servidor;
  4. Acesso o endereço da aplicação;
  5. Crie uma aplicação chamada **pools**;
  6. Registre a aplicação;
  7. Defina uma view;
  8. Defina uma url;
  9. Suba o servidor;
  10. Teste novamente o endereço da aplicação.

26

Django

Introdução:
Instalação, config. inicial de um projeto

Ely – elydasilvamiranda [at] gmail.com

27