APUNTES DE TALLER

07/01/2020

Tema: Django

Estudiante: Alejandro Schmidt Ramírez

1 Experiencia de instalación

Muy sencillo dado a que todos teniamos **pip**. Simplemente se corrió un comando para descargar Django y se descargó el repositorio necesario de Github.

2 Aspectos más relevantes de la tecnología

- Django es una librería de Python.
- Creado por Adrian Holovaty y Simon Willison.
- Es open source y compatible con una gran gama de bases de datos y tecnologías.
- Es muy escalable por su principio de **share nothing**.
- Tiene un sistema de caché para renderizar páginas.
- Trabaja con MTV (model, template y view) y se enfoca en bajo acoplamiento.
- Django es un **Object Relational Model**.
- Aplicaciones que utilizan Django: Youtube, Dropbox, Instagram.

3 Descripción del ejercicio

El ejercicio consistía en construir una pequeña aplicación para crear una lista de tareas por hacer. Primero se crearon varios archivos necesarios para correr el servidor:

- **urls.py**: Maneja las solicitudes al servidor. Simplemente se encarga de definir en una lista las rutas posibles y la función que las maneja.

- **views.py**: Tiene la implementación de las funciones que se llaman desde urls.py.
- **models.py**: Tiene la información del modelo a guardar en la base de datos.
- **forms.py**: Se encarga de definir la estructura del form de HTML para obtener la información del usuario.

4 Evidencias visuales

```
from django.urls import path
from . import views
urlpatterns = [
    path( '' , views.home, name = "home" ),
    path( 'delete/<list_id>' , views.delete, name = "delete" ),
    path( 'cross_off/<list_id>' , views.cross_off, name = "cross_off"
    ↪ ),
    path( 'uncross/<list_id>' , views.uncross, name = "uncross" ),
    path( 'edit/<list_id>' , views.edit, name = "edit" ),
]
```

Listing 1: urls.py

```
from django import forms
from .models import List

class ListForm ( forms . ModelForm ):
    class Meta :
        model = List
        fields = [ "item" , "completed" ]
```

Listing 2: forms.py

```
from django.db import models

class List ( models . Model ):

    item = models.CharField( max_length = 200 )
    completed = models.BooleanField( default = False )

    def __str__ ( self ):
        return self .item + ' | ' + str ( self .completed)
```

Listing 3: models.py

```
from django.shortcuts import render, redirect
from .models import List
from .forms import ListForm
from django.contrib import messages

def home ( request ):
    if request.method == 'POST':
        form = ListForm(request.POST or None )
        if form.is_valid():
```

```
10              form.save()
11              all_items = List.objects.all
12              messages.success(request, ( 'Tarea ha sido agregada a la
    ↪ lista!' ))
13              return render(request, "home.html" ,{ 'all_items'
    ↪ :all_items})
14      else:
15          all_items = List.objects.all
16          return render(request, 'home.html' , { 'all_items' :all_items})
17
18  def delete ( request , list_id ):
19      item = List.objects.get( pk =list_id)
20      item.delete()
21      messages.success(request, ( 'Item Has Been Deleted!' ))
22      return redirect( 'home' )
23
24  def cross_off ( request , list_id ):
25      item = List.objects.get( pk =list_id)
26      item.completed = True
27      item.save()
28      return redirect( 'home' )
29
30  def uncross ( request , list_id ):
31      item = List.objects.get( pk =list_id)
32      item.completed = False
33      item.save()
34      return redirect( 'home' )
35
36  def edit ( request , list_id ):
37      if request.method == 'POST' :
38          item = List.objects.get( pk =list_id)
39          form = ListForm(request.POST or None , instance =item)
40          if form.is_valid():
41              form.save()
42              messages.success(request,( 'Tarea ha sido editada!' ))
43              return redirect( 'home' )
44      else:
45          item = List.objects.get( pk =list_id)
46          return render(request, 'edit.html' , { 'item' :item})
```

Listing 4: views.py