Usługi i Technologie Internetowe
Framework Django

Prowadzący:
Dariusz Marek: dariusz.marek@polsl.pl
Pokój: 906

26 listopada 2019

Spis treści

1 Wprowadzenie

Django to framework umożliwiający tworzenie aplikacji internetowych do Python'a. Aplikacje w Djnago dzielą się na 3 główne części (MTV):

1. Model - część odpowiedzialna za odwozorwanie bazy danych na obiekty
2. Template - część odpowiedzialna za odseparowanie danych modelu od prezentacji
3. View - część opisująca sposób prezentowania danych użytkownikowi, standardowo odwołująca się do szablonów (Template)

W modelu tym, raczej nie używa się podziału MVC (Model, View, Controller) choć jest on zbliżony do działania frameworku Django. Sposób działania MVC został przystępnie przedstawiony na następującej stronie:
https://devloger.pl/mvc-co-to-jest-na-czym-polega-jak-dziala

1.1 Wymagania

Instrukcja powstała dla Django 2.1.2 i Pythona 3.7 do poprawnego działania potrzeba:

1. Python 3.7
2. Dodanego Python'a do Path w zmiennych środowiskowych systemu Windows np.:
   (a) $C:\backslash Python37\backslash$
   (b) $C:\backslash Python37\backslash Scripts\backslash$

1.2 Dokumentacja

https://docs.djangoproject.com/pl/2.2/

1.3 Instalacja modułu

Instalacja modułu Django za pomocą modułu pip do Python'a. W konsoli (cmd) należy wpisać:

```
pip install django
```

1.4 Tworzenie projektu

Tworzenie projetu odbywa się za pomocą konsoli, w której należy wpisać:

```
django-admin startproject <nazwa_projektu>
```

Podstawowe skrypty tworzone podczas inicjalizacji projektu:

1. manage.py - https://docs.djangoproject.com/pl/2.1/ref/django-admin/
2. settings.py - https://docs.djangoproject.com/pl/2.1/topics/settings/
3. urls.py - https://docs.djangoproject.com/pl/2.1/topics/http/urls/
4. wsgi.py - https://docs.djangoproject.com/pl/2.1/howto/deployment/wsgi/

1.5 Serwer deweloperski

Uruchamianie serwera deweloperskiego z projektem odbywa się za pomocą skryptu manage.py wygenerowanego podczas tworzenia projektu:

```
python manage.py runserver
```

2 Tworzenie prostej aplikacji

Tworzenie pierwszego testowego widoku

1. Do skryptu urls.py należy dodać funkcję zwracającą obiekt HttpResponse:

```
from django.http import HttpResponse

def home(request):
  """Renders the home page."""
  return HttpResponse("Home test")
```

2. Zmodyfikować listę urlpatterns

```
urlpatterns = [
  path('',
    home,
    name='home'),
]
```

Dołączenie pliku ze stworzonym widokiem do aplikacji

1. Do folderu z projektem należy dodać folder 'templates' oraz zmodyfikować ustawienia serwera (skrypt settings.py) dodając ścieżkę stworzonego folderu do sekcji TEMPLATES:

```
TEMPLATES = [
  {
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
   'DIRS': [os.path.join(BASE_DIR, 'templates'), ],
```

```
    'APP_DIRS': True,
    'OPTIONS': {...}
  },
]
```

2. Do stworzonego folderu templates należy dodać widok np. index.html

```
Test title: {{ title }}
```

3. Przykładowa modyfikacja funkcji home przesyłająca dane do klienta(funkcja render znajduje się w module django.shortcuts):

```
def home(request):
  return render(
    request,
    'index.html',
    {
      'title':'Home test',
    }
  )
```

Dołączenie szablonu bazowego widoku

1. Należy dołączyć plik z szablonem do folderu z widokami np. base.html

```
To jest baza
<br>
{% block content %}{% endblock %}
```

2. Należy zmodyfikować plik index.html

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

{% block content %}
Test title: {{ title }}
{% endblock %}
```

Proste logowanie wykorzystujące wbudowane moduły Django

1. Należy dołączyć plik z szablonem logowania do folderu z widokami np. login.html

```
{% extends 'base.html' %}

{% block content %}
<h2>Logowanie</h2>
```

```
<formmethod="post">
{% csrf_token %}
{{ form.as_p }}
<buttontype="submit">Login</button>
</form>
{% endblock %}
```

2. Zmodyfikować urlpatterns w skrypcie urls.py

```
from django.urls import path
from django.contrib import admin
from django.shortcuts import render
from django.http import HttpResponse
from django.contrib.auth import views asauth_views

urlpatterns = [
  path('',
    home,
    name='home'),

    path('login/', auth_views.LoginView.as_view(), name='login'),
]
```

3. Oraz dodać/zmodyfikować stronę, na którą zostanie przekierowany użytkownik po zalogowaniu w skrypcie settings.py:

```
LOGIN_REDIRECT_URL = 'home'
LOGOUT_REDIRECT_URL = 'home'
```

Inicjalizacja bazy danych:

```
python manage.py migrate
```

Tworzenie użytkownika:

```
python manage.py createsuperuser
```

Przykładowa aplikacja wyświetlająca informacje o zalogowanym użytkowniku:

1. base.html

```
<!DOCTYPEhtml>
<html>
<head>
<title>{{ title }}</title>
</head>
```

```
<body>
<header>
<h2>First App</h2>
{% if user.is_authenticated %}
Czesc{{ user.username }}!
<br>
Twoj email: {{ user.email }}
<br>
Imie: {{ user.first_name }}
<br>
Nazwisko: {{ user.last_name }}
<br>
<ahref="{% url 'logout' %}">Wyloguj</a>
{% else %}
<ahref="{% url 'login' %}">Zaloguj</a>
{% endif %}
</header>
<hr>
{% block content %}
{% endblock %}
</body>
</html>
```

---

2. urls.py

```
from django.contrib import admin
from django.urls import path, re_path
from django.http import HttpResponse
from django.shortcuts import render
from django.contrib.auth import views asauth_views

def home(request):
  return render(
   request,
    'index.html',
    {
      'title':'Home test',
    }
  )

urlpatterns = [
  path('',
    home,
  name='home'),
```

```
  path('login/', auth_views.LoginView.as_view(), name='login'),
  path('logout', auth_views.LogoutView.as_view(), name='logout'),
]
```

3 Zadanie do wykonania

Wykorzystując przykłady, dokumentację oraz Internet napisz aplikację, która:

1. Będzie posiadała 2 główne widoki:
   (a) Startowy z możliwością logowania
   (b) Z informacjami o zalogowanym użytkowniku umożliwiający wylogowanie użytkownika
2. Będzie umożliwiała zalogowanie użytkownika
3. Będzie posiadała dodanych 2 użytkowników z danymi (nazwa, imię, nazwisko, email) stworzonych za pomocą skryptu/funkcji w Python

   Wskazówka:
   Można stworzyć widok/funkcję (tak jak *home*), w której zostanie dodany użytkownik. Można do tego użyć modułu: *django.contrib.auth.models*, z którego trzeba zaimportować klasę *User*. Tworzenie użytkownika następuje wtedy za pomocą: *User.objects.create_user*()
4. Będzie posiadała dodatkową bazę danych np. Książek z dodanymi 2 elementami, wyświetlającymi się na stronie głównej:

   Wskazówka:

   Można to wykonać za pomocą dodatkowego modułu: *django* − *tables*2
   Tutorial: https://django-tables2.readthedocs.io/en/latest/pages/tutorial.html

   Należy wtedy dodać do sekcji *INSTALLED_APPS* w skrypcie settings.py: projekt w Django oraz moduł *django* − *tables*2.

   Następnie stworzyć skrypt *models.py*, w którym trzeba dodać nową klasę, która zostanie przemapowana na tabelę:

```
from django.db import models

class Musician(models.Model):
  first_name = models.CharField(max_length=50)
  last_name = models.CharField(max_length=50)
```

```
  instrument = models.CharField(max_length=100)
```

Na koniec należy stworzyć i dołożyć tabelę do bazy za pomocą konsoli:

```
python manage.py make migrations <nazwa_projektu>
python manage.py migrate <nazwa_projektu>
```

4 Sprawozdanie

Proszę przesłać na platformę zdalnej edukacji (https://platforma.polsl.pl/):

1. Kody źródłowe zadań
2. Krótkie sprawozdanie z wykonanych zadań
3. Skład sekcji