Zaawansowane Aplikacje Internetowe

- Framework Django
- Podstawowa aplikacja
- Szablony i Moduły
- Dostęp do bazy danych
- Wdrożenie projektu

Opracował: inż. Grzegorz Petri

Zagadnienia

- Struktura projektu i pliki do edycji
- Tworzenie endpointów dostępu
- Proste API z formatem JSON
- API z Serializacją Modeli
- Testowanie API

Struktura aplikacji w projekcie

Aplikacja z API

```
<proj-name>/            1
    manage.py
    <proj-name>/        3
        __init__.py
        settings.py     4
        urls.py         5
        asgi.py
        wsgi.py
    <app-name>/         9
        __init__.py
        admin.py        12
        apps.py         13
        models.py       14
        serializers.py15
        urls.py         16
        views.py        17
```

1 `<proj-name>/` – kontener projektu

3 `<proj-name>/` – główny katalog projektu

4 `<proj-name>/settings.py` – ustawienia projektu

5 `<proj-name>/urls.py` – wskazanie API URL projektu

9 `<app-name>/` – katalog główny aplikacji (tutaj API)

12 admin.py – plik rejestrujący panel administracyjny

13 apps.py – konfiguracja aplikacji

14 models.py – pojedynczy model danych

15 serializers.py – konwersja danych Modelu na JSON

16 urls.py – plik do stworzenia na endpointy API

17 views.py – możliwe do wygenerowania widoki

RESTful API

Wzorce endpointów typu CRUD

- **Przyjmując jako adres projektu:** `http://localhost`
- **Zakładając dla aplikacji katalog (adres):** `MyApi`
- **Endpointy CRUD ▶**

Dowolna nazwa URL

```
localhost/<api-app>/<api-url>/<record-id>
```

```
localhost/api/
localhost/api/getData/
localhost/api/postData/
localhost/api/putData/<int:id>
localhost/api/deleteData/<int:id>
```

Używane akcje HTTP NIE są częścią nazwy

Adresy oferujące API

Tworzenie endpointów

- **Przyjmując jako adres projektu: http://localhost**
- **Zakładając dla aplikacji katalog (adres): MyApi**
- **Planowane Endpointy ▶**

Adresy URL do wpisania w przeglądarce internetowej:

```
localhost/<api-app>/<api-url>
```

```
localhost/api/
localhost/api/test/
localhost/api/list/
localhost/api/data/
```

Aplikacje do obsługi API

Biblioteki CORS, JSON oraz konwersji Modeli

- Zakładając, że katalog projektu to: **MyPrj**
- Zainstaluj w projekcie potrzebne biblioteki ▶

```
user@pc:MyPrj$ pip install <library>
```

```
pip install django-cors-headers
pip install djangorestframework
```

Aplikacje do obsługi API

Biblioteki CORS, JSON oraz konwersji Modeli

- **Zakładając, że nazwa projektu to: MyPrj**
- **Otwórz plik:**
  - ➔ `MyPrj/settings.py`
- **Dopisz do niego kod ►**

MyPrj/settings.py

```
INSTALLED_APPS = [
    'corsheaders',    # nagłówki dla CORS
    'MyApi',          # `aplikacja` z API
    'rest_framework',# obsługa serializacji
    ...
]
CORS_ORIGIN_ALLOW_ALL = True
```

Adres do API projektu

Wskazanie adresu URL dla API Aplikacji

- **Zakładając, że nazwa projektu: MyPrj**
- **Otwórz/utwórz plik:**
  - ➔ `MyPrj/urls.py`
- **Dopisz do niego kod ▶**

MyPrj/urls.py

```
...
from django.urls import include,path


urlpatterns = [
  path('admin', admin.site.urls),
  path('api', include('MyApi.urls')),
]
```

URL, a Aplikacja to 2 różne nazwy

Widoki aplikacji

Główny kod oferujący dane

- **Zakładając, że nazwa aplikacji to: MyApi**
- **Otwórz/utwórz plik:**
  - ➔ `MyApi/views.py`
- **Dopisz do niego kod ▶**

MyApi/views.py

```
from django.shortcut import render
from django.http import HttpResponse
from django.http import JsonResponse

def apiList(request):
  return HttpResponse('<html>')
def apiTest(request):
  return JsonResponse(data,safe=false)
def aList(request):
  return JsonResponse(data)
def apiData(request):
  return JsonResponse()
```

Ścieżki

Mapowanie URL na Widoki

- **Zakładając, że nazwa aplikacji to: MyApi**
- **Utwórz plik:**
  - ➔ `MyApi/urls.py`
- **Dopisz do niego kod ▶**

MyApi/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
  path('', views.apiList),
  path('test', views.apiTest),
  path('list', views.aList),
  path('data', views.apiData), …
```

Widoki aplikacji

Wysyłanie danych ze Stringa

- **Zakładając, że nazwa aplikacji to: MyApi**
- **Otwórz/utwórz plik:**
  - ➔ `MyApi/views.py`
- **Dopisz do niego kod ▶**
- **Widok oferuje tablice w formacie JavaScript**
- **To NIE jest format JSON**

MyApi/views.py

```
from django.shortcut import render
from django.http import JsonResponse

def apiTest(request):
  data = [  // to jest TABLICA nie JSON
    {„klucz1”:”wartość1”},
    {„klucz2”:”wartość2”}
  ]
  return JsonResponse(data, safe=False)
```

Widoki aplikacji

Wysyłanie danych ze Stringa

- **Zakładając, że nazwa aplikacji to: MyApi**
- **Otwórz/utwórz plik:**
  - ➔ `MyApi/views.py`
- **Dopisz do niego kod ►**
- **Widok oferuje dane w formacie JSON**

`MyApi/views.py`

```
from django.shortcut import render
from django.http import JsonResponse

def apiTest(request):
  data = {  ”body”: [ // to jest JSON
    {„klucz1”:”wartość1”},
    {„klucz2”:”wartość2”}
  ] }
  return JsonResponse(data)
```

Widoki aplikacji

Wysyłanie danych z pliku

- **Zakładając, że nazwa aplikacji to: MyApi**
- **Otwórz/utwórz plik:**
  - ➔ `MyApi/views.py`
- **Dopisz do niego kod ▶**

MyApi/views.py

```
from django.shortcut import render
from django.http import HttpResponse
from django.http import JsonResponse
import json

def aList(request):
  with open('MyApi/file.json') as jj:
    data = json.load(jj)
  return JsonResponse(data)
```

Rejestracja aplikacji

Aplikacja w panelu

- **Zakładając, że nazwa aplikacji to: MyApi**
- **Utwórz plik:**
  - ➔ `MyApi/serializers.py`
- **Dopisz do niego kod ▶**

MyApi/serializers.py

```
from rest_framework import serializers

from .models import AppModel
#   ▲ spacja i kropka
class AppSerializer(
    serializers.ModelSerializer):
  class Meta:
    model = AppModel
    fields = ['pola','z','modelu']
```

Widoki aplikacji

Wysyłanie danych z pliku

- **Zakładając, że nazwa aplikacji to: MyApi**
- **Otwórz/utwórz plik:**
  - ➔ `MyApi/views.py`
- **Dopisz do niego kod ▶**
- *** jeżeli trzeba, dopisz argument:** `many=True`
- **** jeżeli trzeba, dopisz:** `safe=False`

`MyApi/views.py`

```
from django.shortcut import render
from django.http import JsonResponse
from .models import AppModel
from .serializers import AppSerializer

def apiData(request):
  model = AppModel.objects.all()
  serial = AppSerializer(model,*)
  return JsonResponse(serial.data,**)
```

```
def questions():
return answer
```