Zaawansowane Aplikacje Internetowe

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

Opracował: inż. Grzegorz Petri

Zagadnienia

- Aplikacja
- Model (danych)
- Pola w modelu danych
- Model (py) a Schemat (sql)
- Szmery bajery (kartkówka)

Umieszczenie aplikacji w projekcie

Aplikacja

Ścieżka do Aplikacji w strukturze Projektu

```
/home/<USER>/<proj-name>/<app-name>
```

3) Załóż projekt

```
$ django-admin startproject <project-name>
```

4) Stwórz aplikację:

```
$ python manage.py startapp <app-name>
```

Poprzez w/w polecenie powstały „puste” pliki (pojemniki) na właściwy kod aplikacji.

Struktura aplikacji w projekcie

Aplikacja

```
<proj-name>/            1
    manage.py

    <proj-name>/        3
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
    <app-name>/         9
        migrations/     10
        __init__.py     11
        admin.py        12
        apps.py         13
        models.py       14
        tests.py        15
        urls.py         16
        views.py        17
```

1 `<proj-name>/` – kontener projektu

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

9 `<app-name>/` – katalog główny aplikacji

10 **migrations/** – katalog śledzenia zmian modeli

11 **__init__.py** – (..) katalog ma być uważany jako pakiet

12 **admin.py** – plik rejestrujący panel administracyjny

13 **apps.py** – konfiguracja aplikacji

14 **models.py** – pojedynczy model danych

15 **tests.py** – możliwe do wykonania testy aplikacji

16 **urls.py** – plik do stworzenia na adresy URL

17 **views.py** – możliwe do wygenerowania widoki

Reprezentacja obiektu

Model

- **Django posiada potężny system ORM (*mapowania obiektowo-relacyjnego*)**
  pole zdefiniowane jest jak atrybut klasy, a te są w relacji ze schematem BD
- **Model reprezentuje kompletny Obiekt w aplikacji**
  klasa (*implementacja modelu*) dziedziczy po klasie `django.db.models.Model`
- **Każdy model posiada pole `id` o unikalnej wartości**
  autoinkrementujące pole jest tworzone automatycznie wraz ze schematem
- **Django posiada duży zbiór pól do użycia w modelach**
  każdy typ pola modelu Django odpowiada typowi pola w Bazie danych

Rejestracja aplikacji

Model

- **Zakładając, że nazwa aplikacji to: MyApp**
- **Otwórz/utwórz plik:**
  - ➔ `MyApp/admin.py`
- **Dopisz do niego kod ►**

`MyApp/admin.py`

```
from django.contrib import admin

from .models import MyApp
#   ▲ spacja
admin.site.register(MyApp)
```

Wzorce adresów dla Aplikacji

Model

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

MyApp/urls.py

```
from django.urls import path

from . import views
urlpatterns = [
    path('',
    views.index,
    name='index'),
]
```

Wzorce adresów Projektu

Model

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

MyPrj/urls.py

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

urlpatterns = [
    path('',
         include('MyApp.urls')),
    path('admin/',
         admin.site.urls),
]
```

Klasa aplikacji

Model

- **Zakładając, że nazwa aplikacji to: MyApp**
- **Otwórz plik:**
  - ➔ `MyApp/models.py`
- **Dopisz do niego kod ►**

MyApp/models.py

```
from django.db import models
# Create your models here.
class FrontPage(models.Model):
  title= models.CharField(max_length=150)
  body = models.TextField()
  timestamp = models.DateTimeField()
```

Klasa aplikacji

Model

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

`MyPrj/settings.py`

```
INSTALLED_APPS = [
    ...
    'MyApp',
]
```

Funkcja widoku

Model

- **Zakładając, że nazwa aplikacji to: MyApp**
- **Otwórz plik:**
  - MyApp/views.py
- **Dopisz do niego kod ►**

MyApp/views.py

```
from django.shortcuts import render
from django.template import loader, Context
from django.http import HttpResponse

# Create your views here.
def index(request):
    return HttpResponse(„Hell no kitty”)
```

Typy pól do użycia w modelu

Pola w modelu danych

AutoField	PK AI
ForeignKey	FK

BigAutoF*	
BigIntegerF*	
BinaryF*	
BooleanF*	
CharF*	
DateF*	
DateTimeF*	
DurationF*	
EmailF*	

FileF*	
FilePathF*	URLF*
FloatF*	UUIDF*
GenericIPAddressF*	
ImageF*	
IntegerF*	
JSONF*	
TextF*	
TimeF*	

Dokumentacja: https://docs.djangoproject.com/en/4.0/ref/models/fields/

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