Zaawansowane Aplikacje Internetowe

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

Opracował: inż. Grzegorz Petri

Zagadnienia

- Widok i Szablon
- Widok a URL
- URLconf i algorytm mapowania
- Widok a Model, Widok a Szablon
- Szablony: katalogi oraz struktura pliku
- Smart code: Skrót dla widoku, Przestrzenie nazw

Czym jest i za co odpowiada

Widok

- **Widok jest typem strony web** w aplikacji Django, który oferuje konkretną **funkcję** oraz posiada konkretny **szablon** strony HTML
- **Każdemu widokowi odpowiada funkcja**
  lub metoda dla widoków opartych na klasach
- **Wybieranie widoku odbywa się przez parsowanie adresu URL**
  Żądany URL (*część po nazwie Domeny*) jest sprawdzany przez *URL dispatcher*
- **Dispatcher `URLconf` mapuje wzorce URL na widoki, np.:**

`http://domena.local`	`/aplikacja/widok/parametry`
ignorowane	*parsowane przez URLconf*

Wzorce adresów Projektu

Widok

- **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('<app>.urls')),
    path('admin/',
         admin.site.urls),
]
```

Mapowanie

Widok a URL

MyApp/urls.py

```python
from django.urls import path
from . import views

urlpatterns = [
    path('',            views.widok1),
    # /MyApp/
    path('widok',       views.widok2),
    # /MyApp/widok/
    path('widok/13',    views.widok3),
    # /MyApp/widok/param
    path('widok/13/x',  views.widok4),
    # /MyApp/widok/param/x
]
```

parsowane przez URLconf

MyApp/views.py

```python
from django.http import HttpResponse


def widok1(request):
    return HttpResponse("Treść 1")

def widok2(request):
    return HttpResponse("Treść 2")

def widok3(request, arg1):
    return HttpResponse("Treść 3")

def widok4(request, arg1):
    return HttpResponse("Treść 4")
```

Wzorce adresów dla Aplikacji

URLconf

Parametry funkcji path():

1) **route** (*trasa – wzorzec URL*)
   URLconf porównuje wzorce URL z tymi z listy od góry do dołu; ignoruje Domenę oraz parametry GET i POST

2) **view** *(widok – funkcja)*
   przekazuje parametr1 do obiektu `HttpRequest` oraz pozostałe parametry ze ścieżki URL jak `$_GET`

3) **kwargs** (*argumenty słownikowe*)

4) **name** (*etykieta*) nazwanie URL-a pozwala dostać się do niego z dowolnego miejsca aplikacji

MyApp/urls.py

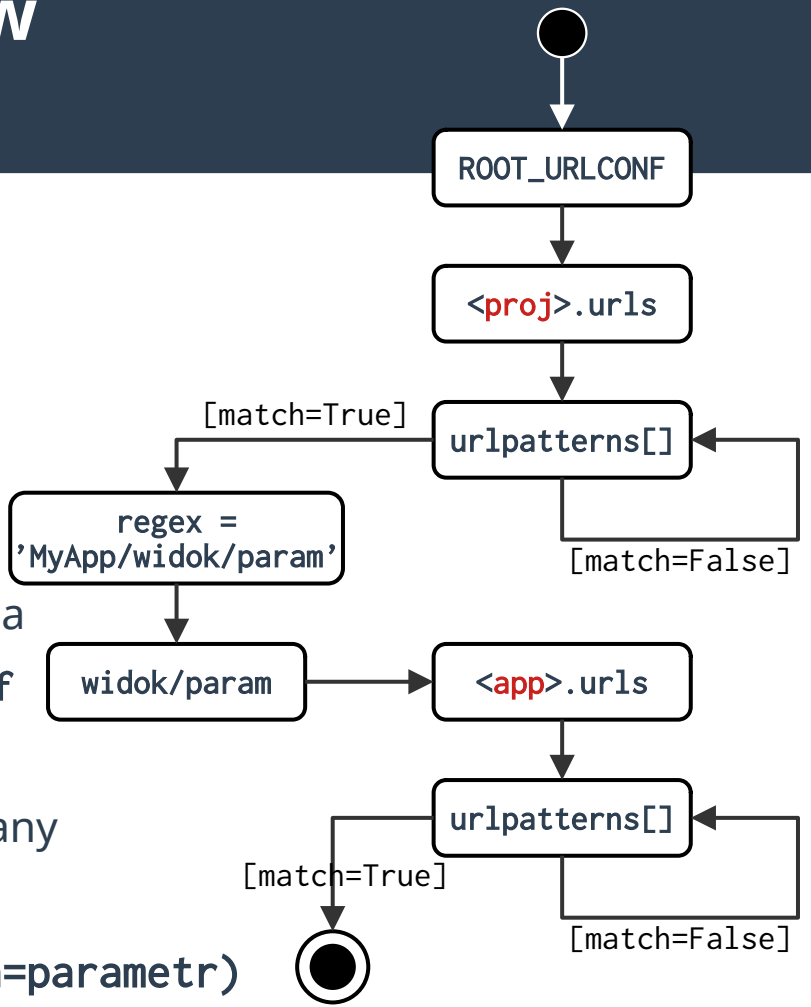
```
from django.urls import path

from . import views
urlpatterns = [
    path(
        'URL',              #1
        views.funkcja,      #2
        kwargs=None,        #3
        name='etykieta'), #4
]
```

Algorytm mapowania adresów

Przykładowy URL: /<app>/widok/parametr/

- `ROOT_URLCONF` wskazuje moduł
- Załadowanie pliku (modułu) `<proj>.urls`
- Znalezienie zmiennej `urlpatterns[]`
- Iterowanie po kolejnych wzorcach
- Znalezienie dopasowania do `<app>/`
- Odcięcie fragmentu `<app>/` z dalszego parsowania
- Wysłanie reszty URL (`widok/parametr/`) do `URLconf`
- Moduł `<app>.urls` przetwarza dalej adres URL
- Po dopasowaniu reszty URL do wzorca wywoływany jest widok:

```
widok(request=<HttpRequest object>, zmienna=parametr)
```



Co za co odpowiada

Widok a Model

MyApp/views.py

```
from django.http import HttpResponse
from django.template import loader
from .models import MyModel

def widok1(request):
    dane = MyModel.objects
    tpl = loader.get_template(
              '<dir>/<file>.html')
    context = {
         'zmienna': dane,
    }
    return HttpResponse(
         tpl.render(context,request)
    )
```

MyApp/models.py

```
from django.db import models

class MyModel(models.Model):
    id = models.ForeignKey(…)
    pole = models.CharField(…)

class OtherModel(models.Model):
    id = models.ForeignKey(…)
    innePole = models.CharField(…)
```

Struktura szablonów w projekcie

Szablony

```
<proj-name>/              1
    manage.py
    <proj-name>/          3
        __init__.py
        settings.py
        urls.py
    <app-name>/           9
        migrations/
        templates/        11
          <dir-name>/     12
            plik.html     13
        __init__.py
        admin.py
        apps.py
        models.py         14
        tests.py
        urls.py           16
        views.py          17
```

1 `<proj-name>/` – kontener projektu

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

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

11 **templates/** – katalog do stworzenia na szablony

12 `<dir-name>/` – katalog na pliki HTML szablonów

13 **plik.html** – konkretny szablon HTML

14 **models.py** – model dostarczający dane do szablonu

16 **urls.py** – plik ze wzorcami adresów URL widoków

17 **views.py** – plik z funkcjami dla widoków

Struktura pliku szablonu

Szablon: <proj>/<app>/templates/<dir>/szablon.html

```
{% if zmienna %}
  <ul>
  {% for iterator in zmienna %}
    <li>{{ iterator.id }}{{ iterator.pole }}</li>
  {% endfor %}
  </ul>
{% else %}
  <p>No polls are available</p>
{% endif %}
```

Legenda:

{% tag %}

- instrukcje / if / for
- wyjście treści
- pobieranie treści z BD

{{ var }}

- zmienne
- listy/tablice
- słowniki
- atrybuty

Widok a Szablon

Szablon

MyApp/views.py

```
from django.http import HttpResponse
from django.template import loader
from .models import MyModel

def widok1(request):
    dane = MyModel.objects
    tpl = loader.get_template(
              '<dir>/<file>.html')
    context = {
         'zmienna': dane,
    }
    return HttpResponse(
         tpl.render(context,request)
    )
```

MyApp/templates/<dir>/<file>.html

```
{% if zmienna %}
  <ul>
  {% for iterator in zmienna %}
    <li>{{ iterator.id }}
        {{ iterator.pole }}</li>
  {% endfor %}
  </ul>
{% else %}
  <p>No polls are available</p>
{% endif %}
```

id – nazwa kolumny z Bazy danych

pole – nazwa kolumny z Bazy danych

Skrót dla widoku

Smart code

MyApp/views.py

```
from django.shortcuts import render
from .models import MyModel

def widok1(request):
    dane = MyModel.objects
    context = {'zmienna': dane}
    return render(
         request,
         '<dir>/<file>.html',
         context)
```

MyApp/templates/<dir>/<file>.html

```
{% if zmienna %}
  <ul>
  {% for iterator in zmienna %}
    <li>{{ iterator.id }}
        {{ iterator.pole }}</li>
  {% endfor %}
  </ul>
{% else %}
  <p>No polls are available</p>
{% endif %}
```

id – nazwa kolumny z Bazy danych

pole – nazwa kolumny z Bazy danych

Przestrzenie nazw

Smart code

MyApp/urls.py

```
from django.urls import path
from . import views
app_name = '<app>'
urlpatterns = [
    path('', views.index, name='etykieta'),
]
```

Elastyczny URL oraz przestrzenie nazw

URL wpisany na sztywno

MyApp/templates/<dir>/<file>.html

```
<li><a href="/<app>/{{ iterator.id }}/">{{ iterator.pole }}</a></li>
# linijkę wyżej zamienić na linijkę niżej
<li><a href="{% url '<app>:etykieta' iterator.id %}/">{{ iterator.pole }}</a></li>
```

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