

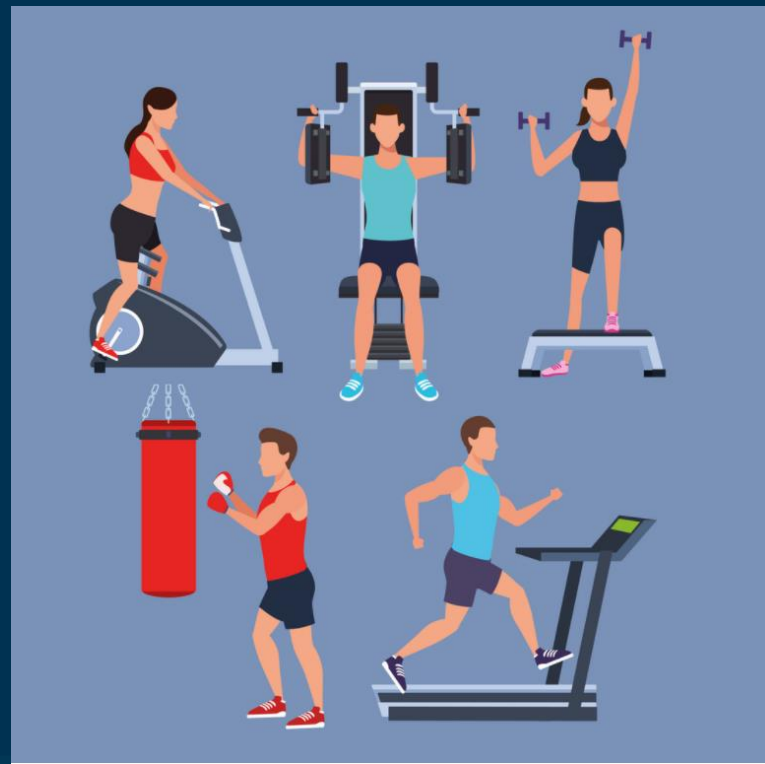


FitForYou

Kamil Sobolak Kacper Zemła

FitForYou

- Wprowadzenie do aplikacji webowych
- Przedstawienie aplikacji
- Czym jest Django ?
- Kilka słów o bazach danych



PROFILE

DIET



WORKOUT



FRIENDS

EXERCISES

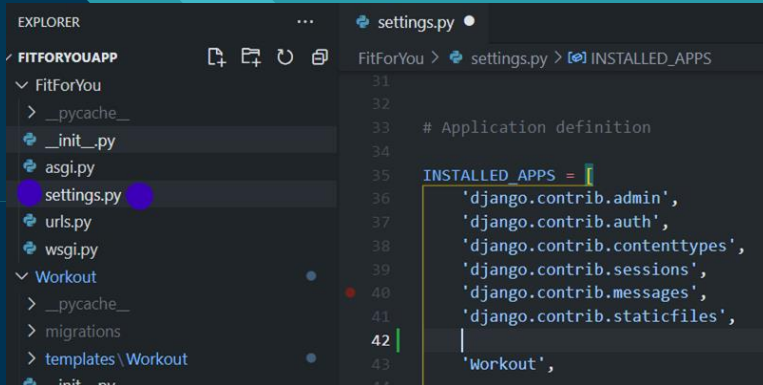
Czym jest DJANGO?

- Jest to framework służący do pisania aplikacji webowych w Pythonie
- Stosunkowo prosty, ale ma wiele ograniczeń (np. rejestracja i logowanie)
- Używa baz danych (domyślnie SQLite)
- Używa wzorca model-template-view



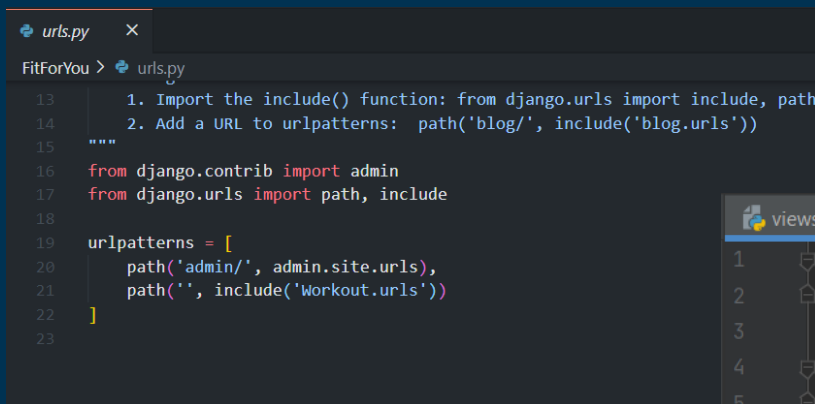
Jak zacząć?

- środowisko obsługujące język python Pycharm, Visual Studio Code
- `pip install django`
- `django-admin startproject nazwa (FitForYou)`
- wejdź do folderu w wierszu poleceń
- `python manage.py startapp nazwa(Workouts)`
- dodanie linii do settings.py (43 screen)
- konfiguracja urls.py, views.py i utworzenie urls.py w folderze aplikacji



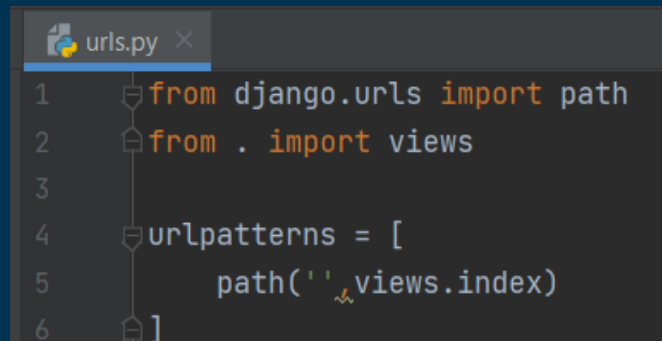
The Explorer sidebar shows a project structure with folders 'FitForYou' and 'Workout'. The 'FitForYou' folder contains files like '__pycache__', '__init__.py', 'asgi.py', 'settings.py', 'urls.py', and 'wsgi.py'. The 'Workout' folder contains '__pycache__', 'migrations', and 'templates\Workout'. The 'settings.py' file is open in the editor, showing the 'INSTALLED_APPS' list. The list includes 'django.contrib.admin', 'django.contrib.auth', 'django.contrib.contenttypes', 'django.contrib.sessions', 'django.contrib.messages', 'django.contrib.staticfiles', and 'Workout'.

```
31
32
33 # Application definition
34
35 INSTALLED_APPS = [
36     'django.contrib.admin',
37     'django.contrib.auth',
38     'django.contrib.contenttypes',
39     'django.contrib.sessions',
40     'django.contrib.messages',
41     'django.contrib.staticfiles',
42
43     'Workout',
44 ]
```



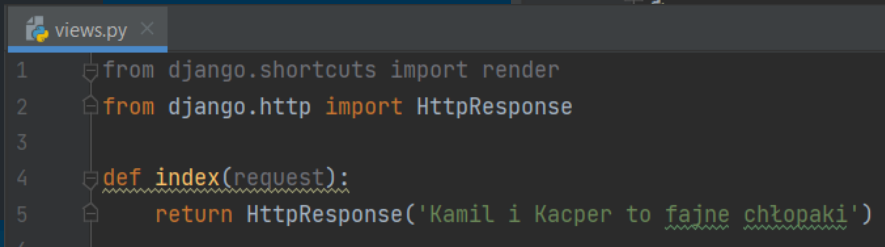
The editor shows the 'urls.py' file for the 'FitForYou' project. The code includes comments and imports for 'include' and 'path' from 'django.urls', and 'admin' from 'django.contrib'. It defines 'urlpatterns' as a list of paths, including 'admin/' and '' (root) which includes 'Workout.urls'.

```
13 1. Import the include() function: from django.urls import include, path
14 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15 """
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('', include('Workout.urls'))
22 ]
23
```



The editor shows the 'urls.py' file for the 'FitForYou' project. The code includes imports for 'path' from 'django.urls' and 'views' from the current directory. It defines 'urlpatterns' as a list of paths, including '' (root) which includes 'views.index'.

```
1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index)
6 ]
```



The editor shows the 'views.py' file for the 'FitForYou' project. The code includes imports for 'render' from 'django.shortcuts' and 'HttpResponse' from 'django.http'. It defines an 'index' function that takes a 'request' and returns an 'HttpResponse' with the text 'Kamil i Kacper to fajne chłopaki'.

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 def index(request):
5     return HttpResponse('Kamil i Kacper to fajne chłopaki')
```

Ważne komendy

- `python manage.py runserver`
- `python manage.py createsuperuser`
- `python manage.py makemigrations`
- `python manage.py migrate`

```
PS C:\Documents\APP_FitForYou\FitForYouApp> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 01, 2021 - 20:55:05
Django version 3.2, using settings 'FitForYou.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
█
```

urls.py

- Są dwa takie pliki, jeden w naszym projekcie, a drugi w naszej aplikacji
- pomaga w przemieszczaniu się między podstronami
- przekazywanie argumentów

```
urlpatterns = [
    path('', views.main, name='main'),
    path('register', views.register, name="register"),
    path('profile', views.profile, name="profile"),
```

```
FitForYou > urls.py > ...
9  ▾ Class-based views
10      1. Add an import: from other_app.views import Home
11      2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
12  ▾ Including another URLconf
13      1. Import the include() function: from django.urls import include, path
14      2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
15      """
16      from django.contrib import admin
17      from django.urls import path, include
18
19  ▾ urlpatterns = [
20      path('admin/', admin.site.urls),
21      path('', include('Workout.urls'))
22  ]
23
```

127.0.0.1:8000/register|

views.py

- zawiera funkcje, które wykonują się po wpisaniu w przeglądarce odpowiedni adres. Funkcja poniżej w przeglądarce załaduje to co znajduje się w pliku main.html (z aplikacji o nazwie Workout)

```
def main(request):  
    return render(request, 'Workout/main.html')
```

- views.py jest połączone z plikiem urls.py, jak widać poniżej jeżeli nasz adres strony będzie domyślny (np. google.com) to z views wykonamy funkcje main

```
urlpatterns = [  
    path('',views.main, name='main'),
```

models.py

- Służy nam do tworzenia odpowiednich modeli w bazach danych
- Każdy model musimy “zarejestrować” w pliku admin.py naszej aplikacji

```
from .models import *  
  
admin.site.register(Task)
```

- Po stworzeniu naszego modelu wykonujemy dwie komendy:

python manage.py makemigrations

python manage.py migrate

```
from django.db import models  
  
# Create your models here.  
  
class Task(models.Model):  
    title = models.CharField(max_length=200)  
    complete = models.BooleanField(default=False)  
    created = models.DateTimeField(auto_now_add=True)  
  
    def __str__(self):  
        return self.title #funkcja ta jest odpowiedzialna
```

forms.py

- Służy nam do tworzenia formularzy na stronie, dzięki temu możemy stworzyć obiekt lub edytować jego pola używając przeglądarki
- class Meta służy nam do tego aby podać z jakiego modelu będzie korzystać (zmienna model) oraz zmienna fields czyli, których pól z modelu chcemy używać (np. title, name itd.)

```
class TaskForm(forms.ModelForm):  
    class Meta: #dla jakiego modelu tw  
        model = Task  
        fields = '__all__' #wszystkie
```

```
<form method="POST" action="/">  
    {% csrf_token %}  
    {{form.title}}  
    <input class="btn btn-info" type="submit"  
</form>
```

```
def index(request):  
    tasks = Task.objects.all()  
  
    if request.method == 'POST':  
        form = TaskForm(request.POST)  
        if form.is_valid():  
            form.save()  
            return redirect('/')  
  
    form = TaskForm()  
    context = {'tasks':tasks, 'form':form}  
    return render(request, 'tasks/list.html', context)
```

filter.py

- problem
- queryset - spryt jest wymagany
- filtrowanie rekordów po przez użytkownika strony

```
settings.py M • filters.py
FitForYou > settings.py > INSTALLED_APPS
35 INSTALLED_APPS = [
36     'django.contrib.admin',
37     'django.contrib.auth',
38     'django.contrib.contenttypes',
39     'django.contrib.sessions',
40     'django.contrib.messages',
41     'django.contrib.staticfiles',
42
43     'Workout',
44
45     'django_filters',
46
47     'django.db.models.utils',
48 ]
49
```

```
import django_filters
from django_filters import CharFilter

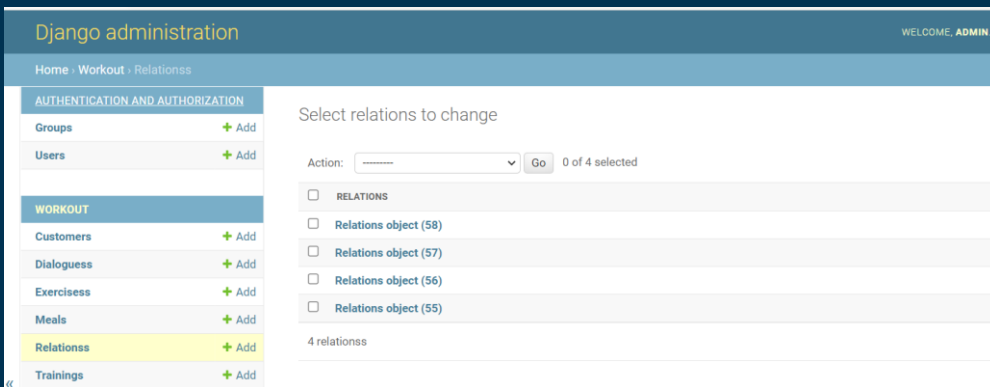
from .models import *

class FriendFilter(django_filters.FilterSet):
    username = CharFilter(field_name = 'username' , lookup_expr = 'icontains')
    class Meta:
        model = Customer
        fields = ['username']

class TextFilter(django_filters.FilterSet):
    text = CharFilter(field_name = 'text' , lookup_expr = 'icontains')
    class Meta:
        model = Dialogues
        fields = ['text']
```

SQLite

- system zarządzania relacyjną bazą danych oraz biblioteka C implementująca taki system
- Baza udostępnia transakcje ACID (przetwarzanie transakcji w bazach danych) oraz implementuje większość standardu SQL 92
- Baza SQLite jest utrzymywana na dysku przy użyciu B-drzew
- Zalety: Idealne dla projektowania a nawet testowania i praca na plikach
- Wady: Nie ma zarządzania użytkownikami i brak możliwości korzystania z dodatkowych usług



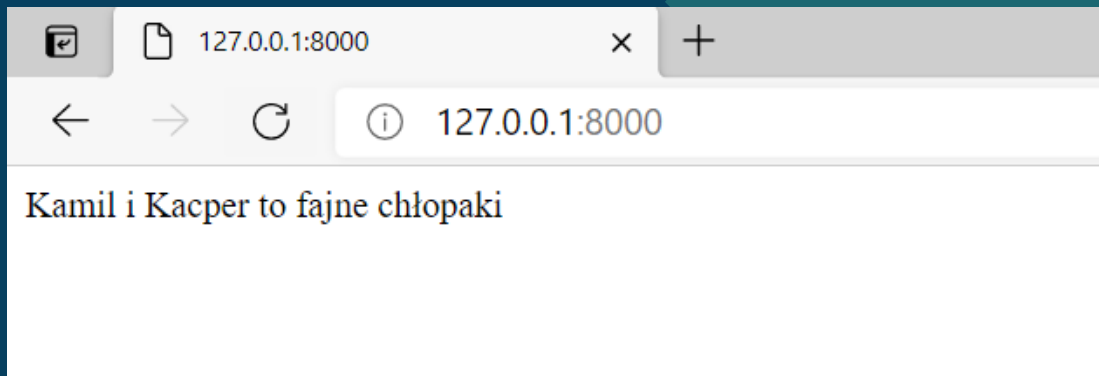
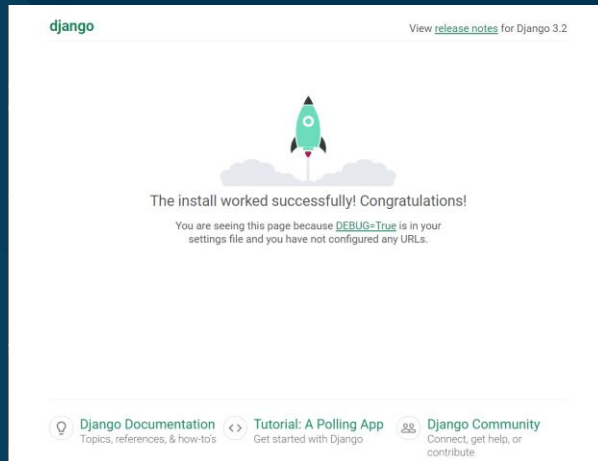
ZADANIE 1

Stwórz samodzielnie od zera projekt. Kieruj się wskazówkami z 9 i 10 slajdu.

Po uruchomieniu serwera powinna wyświetlać się następująca strona.

Następnie tworzymy plik urls.py w folderze aplikacji i konfigurujemy plik views.py.

Po poprawnie wykonanym zadaniu strona powinna wyglądać następująco



ZADANIE 2

Celem będzie stworzenie modelu i stworzenie obiektu w bazie danych.

Stwórz model Exercise, który zawiera pola name, weight, reps, date (slajd 13 i projekt może być pomocny). Ograniczenia każdego pola ustaw według swoich preferencji.

Ustaw aby w bazie danych Exercise było widziane po nazwie (funkcja def __str__(self))

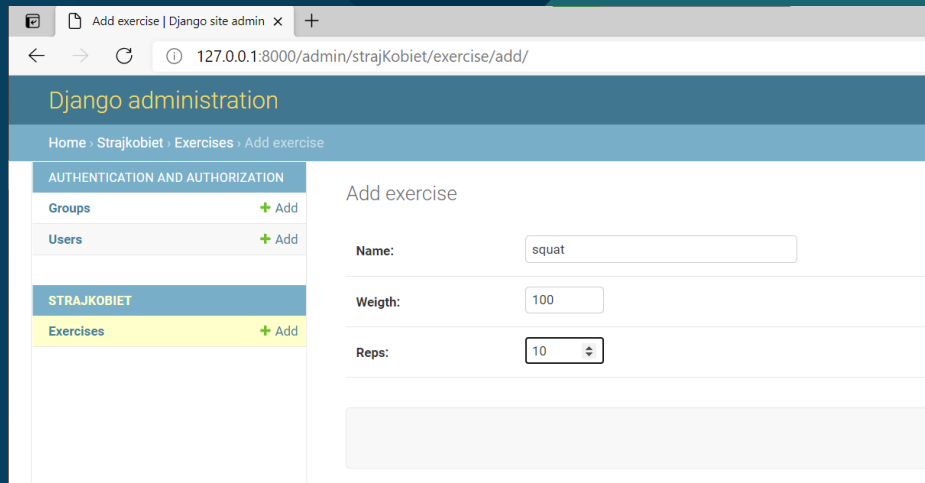
Wykonaj migracje i stwórz superusera (slajd 10)

Skonfiguruj plik admin.py (slajd 13)

Przejdź pod adres <http://127.0.0.1:8000/admin> i zaloguj się

W bazie danych powinno pojawić się pole Exercise.

Dodaj swoje pierwsze ćwiczenie.

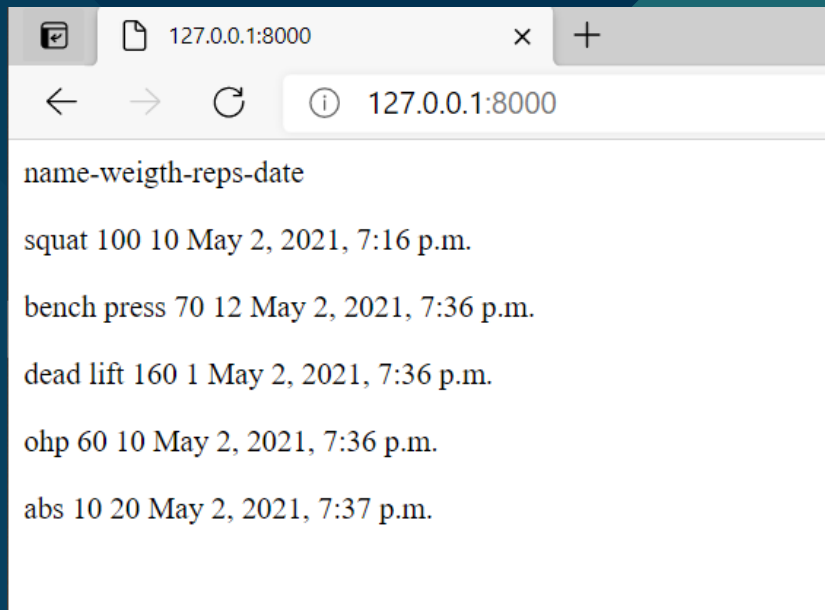


The screenshot shows a web browser window with the Django administration interface. The address bar displays the URL `127.0.0.1:8000/admin/strajKobiet/exercise/add/`. The page title is "Django administration". The breadcrumb trail is "Home > StrajKobiet > Exercises > Add exercise". The left sidebar contains a menu with the following items: "AUTHENTICATION AND AUTHORIZATION" (with sub-items "Groups" and "Users", each with a "+ Add" link), "STRAJKOBIET" (with a sub-item "Exercises" and a "+ Add" link), and "Exercises" (with a "+ Add" link). The main content area is titled "Add exercise" and contains three form fields: "Name:" with the value "squat", "Weigth:" (note the typo) with the value "100", and "Reps:" with the value "10". Below these fields is a large, empty text area.

ZADANIE 3

W bazie danych stwórz 5 ćwiczeń. Twoim celem będzie wyświetlenie ich na stronie. W tym celu utwórz folder `templates` w folderze aplikacji i w nim jeszcze jeden folder o tej samej nazwie co aplikacja (w `views` trzeba zaimportować folder `template`), a w nim plik `html` i w odpowiednim pliku stwórz funkcję, która zwróci ten plik `html` jako widok (wskazówki znajdują się w prezentacji).

Następnie w tej funkcji utwórz zmienną `do`, której przypiszesz wszystkie obiekty `Exercise` (screen ze slajdu 14). Przekaż do funkcji `return render` listę obiektów i wyświetl na stronie Twoje zadania. (Wzoruj się na linii 143 z pliku `exercises.html`, dane wstaw w znacznik `<p>`)



ŹRÓDŁA

- <https://sqlite.org/docs.html>
- <https://stackoverflow.com/questions/tagged/django>
- <https://docs.djangoproject.com/en/3.2/>
- <https://www.youtube.com/channel/UCTZRcDjjkVajGL6wd76UnGg>
- <https://www.youtube.com/channel/UCQtHyVB4O4Nwy1ff5qQnyRw>
- <https://hostovita.pl/blog/porownanie-relacyjnych-systemow-zarzadzania-bazami-danych-sqlite-mysql-postgresql/#zalety-sqlite>
- “Python dla programistów” Paul Deitel Harvey Deitel
- “Django 3. Praktyczne tworzenie aplikacji sieciowych” Mele Antonio