Django - Model danych

Zajęcia Prymus 2020

Agnieszka Rudnicka, rudnicka@agh.edu.pl

Django - Model danych
- Po co nam baza danych?
- "Klasyczne" zarządzanie bazami danych
- Tworzenie tabeli w Django
  - Bonus: podgląd migracji
- Aplikowanie migracji
- Przeglądanie bazy danych przy pomocy DB Browser
- Podsumowanie
- Panel administracyjny
  - Dalsze kroki
  - Dodatkowe materiały

Po co nam baza danych?

Najprościej mówiąc - to kontener w którym zapisane będą informacje o użytkownikach naszej strony, książki, aktualności i wszystkie inne rzeczy które dodamy. Baza danych w naszym przypadku pozwala na przechowanie danych, które będą dodawane przez użytkownika aplikacji, nie przez programistę. My jako programiści jedynie ustalimy reguły wyznaczając modele. Na tych zajęciach skipimy się na bazach relacyjnych, takich jak sqlite3. Warto jednak wiedzieć, że typów baz danych jest wiele, jednym z takich, który znamy z codzienności jest arkusz kalkulacyjny.

W tym konspekcie zaczniemy od modelu przeznaczonego do przechowywania książek.

Zobaczmy jak to wygląda w praktyce...

"Klasyczne" zarządzanie bazami danych

Gdybyśmy nie mieli narzędzi takich jak Django i inne narzędzia, praca z bazami danych (w większości) wymagałaby nauczenia się dodatkowego języka: SQL.

Przykładowe tworzenie tabeli z wykorzystaniem SQL:

```
CREATE TABLE "books_book" (
    "id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
    "title" varchar(100) NOT NULL,
    "short_description" text NOT NULL,
    "published_at" datetime NOT NULL);
COMMIT;
```

Byłby to kod, który trzeba przetestować i gdzieś zapisać. A następnie pamiętać, żeby wykonać na odpowiedniej bazie danych. Gdyby w trakcie projektu wygląd tej tabeli miałby się ulec zmianie (a często się tak dzieje!) wymagane byłyby kolejne skrypty napisane w SQL. A to tylko początek problemów...

Tworzenie tabeli w Django

Tworzenie tabeli z wykorzystaniem Django (w pliku `models.py`):

```
class Book(models.Model)
    title = models.CharField(max_length=100)
    short_description = models.TextField()
    published_at = models.DateTimeField()
```

Poniższym poleceniem możemy wygenerować migrację. Migracja w tym przypadku to skrypt w Pythonie, który pozwala na utworzenie potrzebnych tabel w bazie danych, która jest podpięta do aplikacji. W naszym przypadku tabele zostaną stworzone w bazie `sb.sqlite3`, która znajduje się domyślnie w głównym katalogu całego projektu/repozytorium kodu.

```
manage.py makemigrations
```

Po wykonaniu powyższego polecenia w katalogu `migrations` pojawi się nowy plik nazwany `0001_init.py`. Migracje są automatycznie numerowane, natomiast ich nazwę można ustawić (`manage.py makemigrations -n nazwa` wygeneruje `numer_nazwa.py`).

Na pierwszy rzut oka może się wydawać, że kodu/pracy wcale nie jest mniej. Jednak kluczowe zyski z korzystania z frameworka to:

- nie trzeba się uczyć dodatkowego języka
- framework jest dobrze przetestowany przez twórców i użytkowników, więc możemy założyć, że działa
- przy każdej zmianie w modelu wystarczy uruchomić polecenie w konsoli, a SQL zostanie wygenerowany automatycznie - nie musimy się zastanawiać co się zmieniło

tl;dr: Oszczędzamy czas dzięki gotowym narzędziom :)

Bonus: podgląd migracji

Jeśli chcielibyśmy jedynie podejrzeć co zostanie wygenerowane możemy wykorzystać dodatkowe flagi:
`--dry-run -v 3`
Pierwsza część pozwala na "wykonanie na sucho", czyli cała maszyneria zostanie uruchomiona, ale migracja nie zostanie zapisana na dysku/stworzona. `-v 3` natomiast pozwala na wypisanie wszystkich szczegółów procesu generowania migracji a także samej migracji. Wynik działania tego polecenia możemy zobaczyć poniżej:

```
manage.py makemigrations --dry-run -v 3
```

```
Migrations for 'books':
  books\migrations\0001_initial.py
    - Create model Book
Full migrations file '0001_initial.py':
# Generated by Django 3.0.5

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Book',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
                ('title', models.CharField(max_length=100)),
                ('short_description', models.TextField(max_length=500)),
                ('published_at', models.DateTimeField()),
            ],
        ),
    ]
```

Jak widzimy, najpierw narzędzie poinformowało nas, że tworzona zostaje migracja dla aplikacji "books" a następnie w terminalu wypisana została cała migracja (ten krótki kawałek kodu w Pythonie począwszy od `# Generated by Django 3.0.5` do samego końca).

Aplikowanie migracji

Teraz kiedy mamy już napisany podstawowy model oraz wygenerowaną migrację trzeba ją zaaplikować na bazę danych.

```
manage.py migrate
```

Utworzy to odpowiednią tabele na nasze książki w bazie danych.

Przykładowy output:

```
Operations to perform:
  Apply all migrations: admin, auth, books, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
```

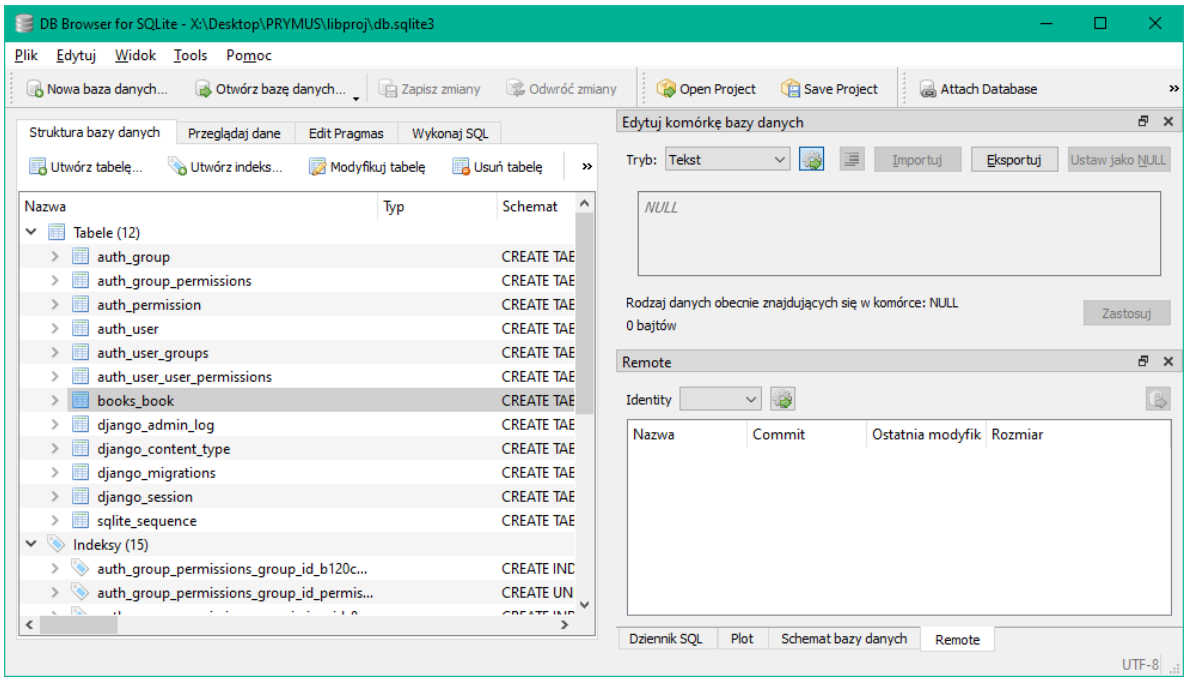
```
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying books.0001_initial... OK
  Applying sessions.0001_initial... OK
```

W drugiej od dołu linijce widnieje nasza migracja! Ona wraz z innymi została zaaplikowana.

Przeglądanie bazy danych przy pomocy DB Browser

Do przeglądania bazy danych SQLite można wykorzystać https://sqlitebrowser.org/

Jest to mały darmowy program przedstawiony na screenie poniżej.



Podsumowanie

Reasumując, aby przechowywać informacje w bazie danych musimy:

1. Opisać w Django (plik `models.py`) jak ma się nazywać model, jakie ma mieć pola oraz jakie właściwości mają te pola posiadać. (np. pole "data_urodzenia", typ: data, wymagane)
2. Wygenerować migrację (polecenie: `manage.py makemigrations`)
3. Zaaplikować migrację na bazie danych, co spowoduje utworzenie odpowiednich tabel lub wprowadzenie zmian w istniejących. (polecenie: `manage.py migrate`)

Panel administracyjny

Najszybszym (i dla niektórych najwygodniejszym) sposobem, żeby dodać nowe książki będzie wykorzystanie panelu administracyjnego Django. Jest to narzędzie dostarczone z frameworkiem, domyślnie włączone.

Aby z niego skorzystać musimy zarejestrować nasz model `Book` jako obsługiwany przez panel administracyjny. W tym celu otwórzmy plik `admin.py` znajdujący się w katalog aplikacji (`books/admin.py`).

Najdziemy tutaj jedynie jeden import i komentarz. Dopiszmy więc wymagany kod:

```
from django.contrib import admin

# Register your models here.
from books.models import Book  # NEW

admin.site.register(Book)  # NEW
```

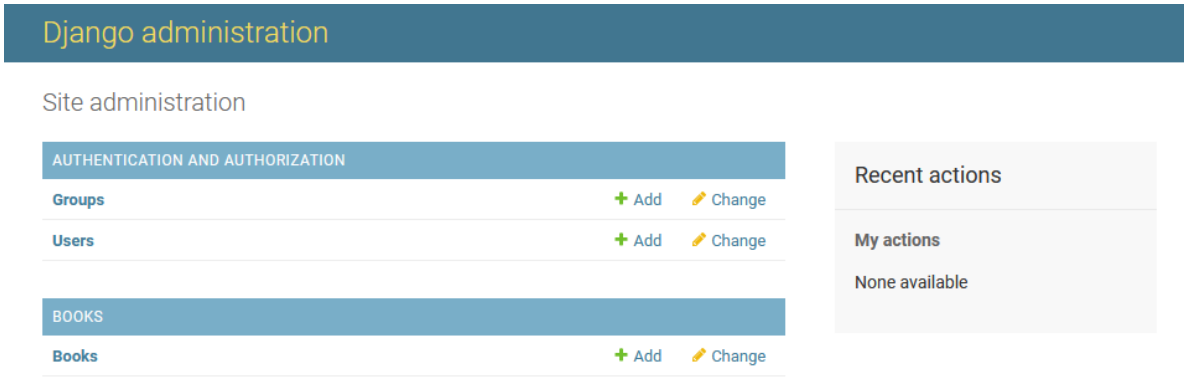
W ten sposób aplikacja panelu administracyjnego będzie wiedziała, że istnieje model `Book` i ma być on zarządzany przez panel.

Uruchommy więc aplikację i przejdźmy pod http://127.0.0.1:8000/admin/

Przypomnienie:

```
# polecenie do uruchomienia
manage.py runserver

# polecenie do utworzenia super-użytkownika/admina
manage.py createsuperuser
```



Dalsze kroki

- Dodać kilka książek
- Otworzyć bazę danych za pomocą programu DB Browser i przeglądnąć dane w tabeli `books_book`
- Konfiguracja etykiety dla obiektów na liście
- Dodanie kolejnych pól do modelu

Dodatkowe materiały

- Spis dostępnych pół, które można dodać do modelu: https://docs.djangoproject.com/en/3.0/ref/models/fields/#field-types
- Django Cheat Sheet, czyli spis często używanych poleceń i kawałków kodu: https://github.com/lucrae/django-cheat-sheet
- Czym są modele, Django Book: https://djangobook.com/mdj2-models/
- Wytłumaczenie czym jest model dla osób mniej zaznajomionych z programowaniem: https://tutorial.djangogirls.org/pl/django_models/