Покрокова інструкція: Створення Django-додатку для анкетування

Цей посібник допоможе вам створити простий, але функціональний веб-додаток, який збиратиме інформацію від користувачів через форму. Ви навчитеся, як пов'язати **Django-форми**, **шаблони**, **представлення** та **маршрутизацію** в один працюючий проєкт.

Крок 1: Підготовка проєкту та додатку

1. **Створіть Django-проєкт:** Якщо ви ще не зробили цього, відкрийте термінал і виконайте команду:

```
django-admin startproject myproject
```

2. **Перейдіть в директорію проєкту:**

```
cd myproject
```

3. **Створіть додаток:**

```
python manage.py startapp survey_app
```

4. **Зареєструйте додаток:** Відкрийте файл myproject/settings.py і додайте 'survey_app' до списку INSTALLED_APPS.

```
INSTALLED_APPS = [
    # ...
    'survey_app',
]
```

Крок 2: Створення структури шаблонів та статичних файлів

Щоб Django міг знайти наші HTML-файли та CSS, нам потрібно створити відповідні директорії.

1. У корені додатка survey_app створіть папку templates.

2. Всередині templates створіть ще одну папку з назвою додатка: survey_app. Таким чином, повний шлях буде survey_app/templates/survey_app.
3. Тут будуть знаходитись наші шаблони: base.html, index.html, survey.html, results.html.
4. У корені додатка survey_app також створіть папку static.
5. Всередині static створіть папку css і помістіть туди файл style.css.

Ваша структура проєкту повинна виглядати приблизно так:

```
myproject/
├── myproject/
│   ├── settings.py
│   └── urls.py
└── survey_app/
    ├── __init__.py
    ├── forms.py
    ├── urls.py
    ├── views.py
    ├── templates/
    │   └── survey_app/
    │       ├── base.html
    │       ├── index.html
    │       ├── results.html
    │       └── survey.html
    └── static/
        └── css/
            └── style.css
```

Крок 3: Написання коду для форм, представлень та маршрутизації

Тепер, коли структура готова, час написати код, який оживить наш додаток.

1. **Визначення форми (survey_app/forms.py):** У цьому файлі ми створюємо клас SurveyForm, який визначає всі поля анкети. Django автоматично перетворить їх на HTML-елементи.
2. **Налаштування представлень (survey_app/views.py):** Це «мозок» нашого додатку. Кожна функція-представлення відповідає за логіку однієї зі сторінок.
   - index_view: просто відображає головну сторінку.
   - survey_view: обробляє як GET-запити (показує порожню форму), так і POST-запити (перевіряє, валідує і перенаправляє).
   - results_view: отримує дані з сесії, щоб відобразити результати.
3. **Налаштування маршрутизації (survey_app/urls.py):** Цей файл пов'язує URL-адреси з відповідними функціями-представленнями. Наприклад, path('survey/', views.survey_view, ...) говорить Django, що за адресою /survey/ потрібно викликати функцію survey_view.

4. **Включення маршрутів додатку в головний файл маршрутизації:** Відкрийте myproject/urls.py і додайте рядок path('', include('survey_app.urls')). Це дозволить Django «побачити» маршрути, які ви визначили в додатку survey_app.

Детальніше про поля форми

У нашому файлі forms.py ми визначаємо п'ять різних полів, які збирають різну інформацію від користувача.

- **name = forms.CharField()**: Це стандартне текстове поле. Воно ідеально підходить для збору коротких рядків, таких як ім'я.
- **favorite_os = forms.ChoiceField()**: Це поле генерує випадаючий список. Ви використовуєте choices для визначення варіантів вибору. Django відображає його як HTML-тег <select>.
- **accept_terms = forms.BooleanField()**: Це поле створює прапорець (checkbox). Воно використовується для булевих значень (True або False). Параметр required=True робить його обов'язковим для позначення, інакше форма буде невалідною.
- **birth_date = forms.DateField()**: Це поле для збору дати. За допомогою widget=forms.DateInput(attrs={'type': 'date'}) ми вказуємо Django, щоб він використовував HTML5-віджет для вибору дати, що зручно для користувача.
- **interests = forms.MultipleChoiceField()**: Це поле дозволяє користувачу вибрати кілька варіантів. Ми використовуємо widget=forms.CheckboxSelectMultiple, щоб показати варіанти як набір окремих прапорців, а не випадаючий список.

Крок 4: Створення та інтеграція шаблонів

Цей крок присвячений HTML-файлам, які будуть відображати наші сторінки. Ми будемо використовувати успадкування шаблонів (extends) для повторного використання загального макету.

1. **base.html:** Це наш базовий шаблон, який містить загальну структуру сторінки (хедер з навігацією, тіло, футер). Він має містити:
   - **У хедері (<header>)**: Навігаційну панель з посиланнями на {% url 'index' %} (Головна) та {% url 'survey' %} (Анкета). Зверніть увагу, що ми використовуємо тег {% url %}, а не жорсткі посилання /, щоб забезпечити гнучкість.
   - **У футері (<footer>)**: Посилання та іншу інформацію, яка є на кожній сторінці. У нашому випадку це просто текст про авторські права.
   - **Блок контенту ({% block content %}{% endblock %})**: Тут буде розміщуватися унікальний вміст кожної сторінки.
   - **Підключення стилів**: Не забудьте підключити CSS-файл за допомогою {% load static %} та <link rel="stylesheet" href="{% static 'css/style.css' %}">.
2. **index.html та results.html:** Ці шаблони успадковують base.html і просто заповнюють блок content своїм унікальним вмістом. **Обов'язково додайте рядок {% extends "survey_app/base.html" %} на початку файлу.**
3. **survey.html:** Найважливіший шаблон. Він також успадковує base.html і містить HTML-форму. Завдяки ручному відображенню полів ({{ form.name }}, {{

form.favorite_os }} і т.д.), ви маєте повний контроль над тим, як вони відображаються, і можете додати власні класи, мітки та стилізовані повідомлення про помилки.

Крок 5: Додавання стилів

Для візуального оформлення ми створили файл style.css. Додайте цей CSS-код у файл survey_app/static/css/style.css і не забудьте додати тег <link> у base.html для його підключення.

Крок 6: Запуск проєкту

Після того, як ви створили всі файли та додали код, ви можете запустити сервер розробки, щоб побачити свій додаток у дії.

```
python manage.py runserver
```