Конспект **Django**

Создаем убийцу **StackOverFlow**

Тексты, написанных красным цветом считается, что эти тексты попадут в тест.

Відношення один до багатьох (**one to many**) передбачає, що один запис моделі може мати багато інших записів моделі, пов’язаних із собою. Наприклад, із записом моделі меню може бути пов’язано багато записів моделі елемента, але елемент належить до одного запису меню.

Також є **one to one** і **many to many.**

Для начало, мы создаем проект с названием «**StackOverflow**», приложение с названием «**forum**», регистрируем приложение и так далее, это мы должны были написать это в прошлых конспектах.

Теперь в templates создаем create\_question.html и пишем сюда:

<**h1**>Створити запитання</**h1**>

**<form method=”post”>**

**{% csrf\_token %}**

***Django має тег {% csrf\_token %}, який реалізовано, щоб уникнути зловмисних атак.***

**<p><input type=”text” name=”author” required placeholder=”Ваше ім’я”></p>**

**<p><input type=”text” name=”topic” required placeholder=”Тема запитання”></p>**

**<p><textarea name=”text” placeholder=”Ваше запитання” required></p>**

**<button>Створити<button>**

**</form> *(post отправляет запрос, а get принимает запрос)***

В **models.py** создаем класс **Question(models.Model)** и пишем внутри:

**author = models.CharField(max\_length=255)**

**topic\_question = models.CharField(max\_length=255)**

**text = models.TextField()**

Создаем и проводим миграцию этой модели через команду в терминале python **manage.py makemigrations** и **python manage.py migrate**.

В **views.py** пишем:

**def create\_question(request):**

**return render(request,”create\_question.html”)**

В **urls.py** импортируем **views.py** и пишем в списке **ulrpatterns:**

**path(‘create\_question/’,create\_question),**

Давайте запустим сервер, чтобы создалась база данных через **python manage.py runserver**.

В **create\_question** импортируем **models.py** и пишем:

**context = dict()**

**if request.method == “POST”:  
 author = request.POST.get(“author”)**

**topic\_question = request.POST.get(“topic\_question”)**

**text = request.POST.get(“text”)**

**Question.objects.create(title = title, content = content, author = author)**

**context[“message”] = “Запитання створено”**

**return render(request,”create\_question.html”,context=context)**

В файле **create\_question.html** перед **form** пишем:

**<p style=”color: green;”>{{ message }}</p>**

В **views.py** создаем функцию **show\_question(request)** и пишем в нем:

**return render(request,’question.html’)**

В **templates** создаем ещё один файл под названием **question.html** и пишем

**<h1>{{ question.topic\_question }}</h1>**

**<p>{{ question.text }}</p>**

**<h5>Автор:{{ question.author }}</h5>**

В **views.py** импортируем из **django.shortscust get\_objects\_or\_404** в функции **show\_question(request,question\_pk)** и пишем:

**context = {**

**"question":get\_object\_or\_404(Question,pk=question\_pk)**

**}**

В **urls.py** пишем:

**path("question/<question\_pk>",show\_question)**

**get\_object\_or\_404** як і слід із назви повертає об'єкт, а якщо він не знайдений, викидає ексепшн 404 сторінка не знайдена.

В **models.py** пишем ещё один класс:

**class Answer(models.Model):  
 author = models.CharField(max\_length=255)**

**text = models.TextField()**

**question = models.ForeignKey(“Question”,on\_delete=models.PROTECT)**

Зовнішній ключ (**ForeignKey**) — це процес, за допомогою якого поля однієї таблиці можна гнучко використовувати в іншій таблиці.

КАСКАД(**models.CASCADE**) – якщо для аргументу **on\_delete** встановлено значення **cascade**, видалення об’єкта, на який посилається, матиме значний вплив на об’єкти, на які посилається. Це означає, що коли об’єкт, на який посилається, буде видалено з бази даних, усі записи цього об’єкта також буде видалено з усієї бази даних.

**PROTECT** є прямою протилежністю каскадного параметра, тут, якщо є вплив на фактичний об’єкт, тоді всі екземпляри даних на об’єкті, на який посилаються, не видаляються. Отже, видалення не відбувається. Це забезпечує захист даних перед видаленням об’єкта, на який посилається.

Далее просто создаем и проводим миграции.

В **question.html** пишем:

**<h3>Додайте вашу відповідь</h3>**  
**<form method=”post”>**

**{% csrf\_token %}**

**<p><input type=”text” name=”author” required placeholder=”Ваше ім’я”></p>**

**<p><textarea name=”text” placeholder=”Ваше відповідь” required></p>**

**<button>Створити<button>**

**</form>**

В **views.py** импортируем из **models.py Answer** и в функции **show\_question** пишем: **if request.method == “POST”:  
 author = request.POST.get(“author”)**

**text = request.POST.get(“text”)**

**Answer.objects.create(author=author,text=text,question\_id=question\_pk)**

В **context** из **show\_question** пишем:

**„list\_answers“:Answer.objects.filter(question\_id=question\_pk)**

В **question.html** пишем перед form пишем:

**<br><br><br>**

**<h2>Відповіді:</h2>**

**{% for answer in list\_answers %}**

**<h4>{{ answer.author }}</h4>**

**<p>{{ answer.text }}</p>**

**<hr>**

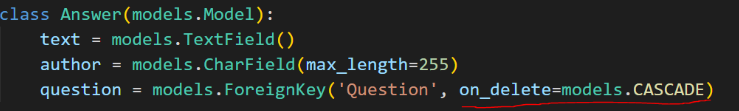
**<hr>** елемент служить для тематичного розділення абзаців.

**{% endfor %}**

Тест

1.Який клас використовуємо замість \_\_\_, щоб створити зв'язок моделей "Один до багатьох" : **ForeignKey**

2. Клієнт зробив запит за адресою [cute-animals.com/cats](http://cute-animals.com/cats). У відповідь клієнт отримав від веб-сервера сторінку з котиками. Тип http-запиту, який надіслав клієнт на веб-сервер - це ... **GET**

3. На зображенні параметр on\_delete вказує, що після видалення запитання пов'язані з ним відповіді ... **будуть також видалені.**

4. Клієнт зареєструвався на сайті, заповнивши форму та надіславши свій логін та пароль на веб-сервер. Тип http-запиту, який надіслав клієнт на веб-сервер - це ... **POST**

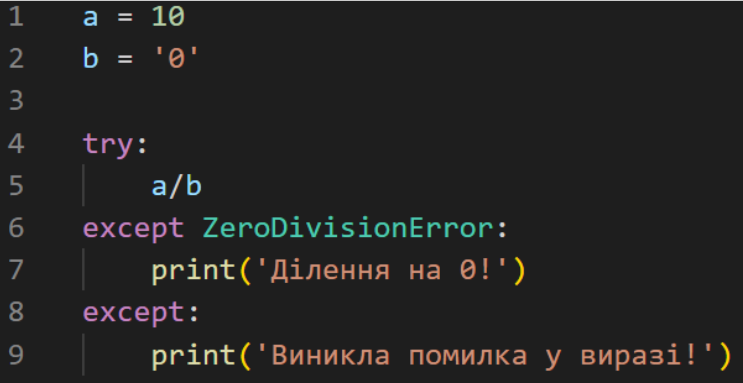
5. У якому з варіантів правильно сформована динамsчна url адреса? **question/<question\_pk>**

6. Кожна форма на веб-сайті обов‘язково має містити у собі певний токен, що підтверджує оригінальність цієї форми. Про який токен йде річ? **csrf\_token**

7. Впишіть ім'я функції, що намагається знайти запис у моделі за вказаними параметрами, а у разі, якщо такого запису не існує - генерує помилку 404: **get\_object\_or\_404**

8. Який метод автоматично переспрямовує (перекидує) користувача на іншу сторінку?: **redirect**

9. Коли користувач заходить у свій акаунт на сайті, Django генерує спеціальний "ключ", який прив'язується до користувача і є його ідентифікатором. "Ключ", про який іде річ - це ... **session\_id**

10. Що виведе програма? : **Виникла помилка у виразі!**