Разработка программного обеспечения на языке Python

Обзорная панель

Мои курсы

<u>Разработка ПО на языке Python</u> <u>Веб-программирование на Python</u>

Лекция 4. Основы баз данных, СУБД

Лекция 4. Основы баз данных, СУБД

Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

Операции с моделями в проекте django



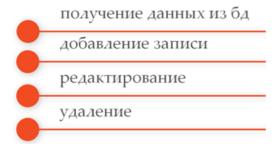
Шаблон формы для отправления цанных на сервер

rm method="POST" action="create/">
{% csrf_token %} <label>Segure wmm</label><pr>

00:00 / 04:31

R

В данной теме рассмотрим способы вывода данных из базы для отображения их на страницах веб-приложении django, а также заполнения этих таблиц пользователем.



Рассмотрим создание и вывод объектов модели на примере. В файле models.py определена модель Student и связанная с ней модель Group.

```
models.py ×
       from django.db import models
1
2
3
     class Group(models.Model):
           name = models.CharField(max_length=5)
6
7
      dclass Student(models.Model):
8
           name = models.CharField(max_length=20)
9
           age = models.IntegerField()
10
           group = models.ForeignKey(Group, on_delete=models.CASCADE)
11
```

В файле **views.py** пропишем два представления для получения данных из бд и для сохранения данных. В функции index() получаем все данные из таблиц студент и группа с помощью метода objects.all() и передаем их в шаблон index.html.

```
🐉 views.py 🗡
      from django.shortcuts import render
 1
       from django.http import HttpResponse, HttpResponseRedirect
 2
      ☆from .models import *
 3
 4
 5
       # получение данных из бд
 6
     def index(request):
 7
           students = Student.objects.all()
 9
           groups = Group.objects.all()
10
           return render(request, "index.html", {"students": students, 'groups':
```

В функции create() создаем объект класса студент. Затем получаем данные с формы из запроса типа POST и записываем их в качестве значения полей нового объекта.

```
% views.py 🗡
13
       # сохранение данных в бд
      def create(request):
14
           if request.method == "POST":
15
               student = Student()
16
17
               student.name = request.POST.get("name")
               student.age = request.POST.get("age")
18
19
               # получение id группы с формы
20
               group_id = request.POST.get("group")
21
22
               # получение записи связанной таблицы по id
23
               group = Group.objects.get(id=group_id)
24
25
               student.group = group
26
27
               student.save()
           return HttpResponseRedirect("/")
28
```

Если в таблице есть внешний ключ, связывающий ее с другой таблицей, то при создании объекта требуется в качестве значения поля задать объект класса связанной таблицы. Из определения таблиц мы видим, что Student связана с таблицей group. То есть в нашем примере значением поля group таблицы Student будет являться объект класса Group. Его значение мы получаем с помощью запроса к таблице группы по заданному id. Далее сохраняем данные с помощью метода save() и выполняем переадресацию на корень веб-сайта (то есть на функцию index).

В папке templates определим шаблон index.html, который будет выводить данные на веб-страницу.

```
# index.html ×
            <!DOCTYPE html>
 1
 2
         -<head>
 3
                  <meta charset="utf-8" />
 4
                  <title>Деканат</title>
 5
 6
         d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
d
<p
         d<body class="container">
 7
                  <form method="POST" action="create/"...>
 8
29
                  {% if students.count > 0 %}
30
                  <h2>Список студентов</h2>
                  31
                         HomepVMmqBospactFpynna
32
                         {% for student in students %}
                         {{ student.id }}{{ student.name }}
34
35
                                {{ student.age }}{{ student.group.name }}
                         {% endfor %}
36
                  37
                  {% endif %}
         </body>
39
         △</html>
40
```

В шаблоне определена таблица, в которую выводятся значения из списка students, переданного ранее из представления. Для этого используем встроенный шаблонизатор django, позволяющий организовать цикл по списку объектов. Каждый элемент в цикле представляет собой объект класса Студент, к свойствам которого мы можем обратиться и записать их в отдельную строку таблицы.

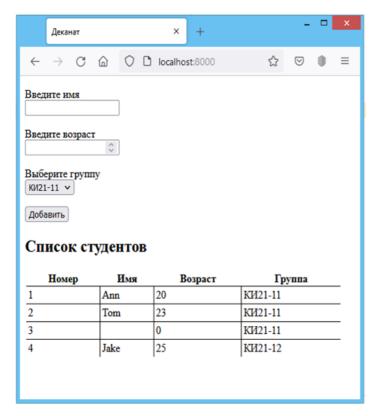
В начале шаблона определена также форма для добавления данных, которые потом будет получать функция create в POST-запросе.

В форме в параметре action указан маршрут из списка urls, который выполняться при отправлении данных с формы. Форма содержит описание всех атрибутов модели, для создания объекта которой она предназначена. В нашем примере – для класса студент. Атрибут пате каждого поля формы содержит имя переменной, с которым она отправляется на сервер. При этом в качестве атрибута group нашей модели выводится список существующих записей таблицы Группа.

И также в файле urls.py свяжем маршруты с представлениями, которые будут выполняться при переходе по ним.

Добавим стили для отображения границ таблицы.

Запустим проект и обратимся к приложению в браузере. Добавим несколько объектов через форму. И после каждого добавления мы увидим, что в таблице появляются новые записи. В случае, если требуется добавить функции редактирование и удаление объектов модели, действия будут аналогичными: потребуется добавить новые функции представлений, шаблоны и маршруты.



В данном примере форму для ввода данных мы создали в явном виде, задав поля формы и значения, которые будут передаваться. При этом формы могут быть созданы как отдельные объекты, в том числе на основе моделей. Для этого требуется в файл forms.py создать класс и указать, что в качестве метаданных для формы будет использоваться модель (в нашем примере, Student).

```
ち forms.py 🗡
1
     from django import forms
2
       from django.forms import ModelForm
     dfrom .models import *
3
 4
 5
     class StudentForm(ModelForm):
6
           class Meta:
7
               model = Student
8
 9
10
               fields = ['name', 'age', 'group']
               labels = {
11
                    'name': 'Имя', 'age': 'Возраст', 'group': 'Группа'
12
13
```

В случае, когда форма описана в виде отдельного объекта, она передается в представлении как параметр, который потом размещается в шаблоне.

index.html

```
<form method="POST" action="create/">
    {% csrf_token %}
    {{form}}
    <input type="submit" value="Добавить" >
</form>
```

views.py

```
# получение данных из бд

def index(request):
    students = Student.objects.all()
    form = StudentForm()
    return render(request, "index.html", {"students": students, 'form': form})
```

Получение данных в представлении выглядит следующим образом. В данном случае в объект класса формы передаются все данные, полученные от пользователя. Далее форма проверяется на валидность - корректность введенных данных. В случае успешной проверки - можно сохранить запись в таблицу с помощью вызова метода save() у объекта формы.

```
# сохранение данных в бд

def create(request):
    if request.method == 'POST':
        formset = StudentForm(request.POST)
        if formset.is_valid():
            formset.save()
        return HttpResponseRedirect("/")
```

Подведем итоги, в данной теме мы рассмотрели добавление в веб-приложение форм для ввода пользовательских данных, которые потом записываются в базу. А также способы вывода данных из таблиц для отображения их на страницах веб-приложения.

Вопросы

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

Перейти на...

•	Задание	5.	Добавление	шаблонов	страниц	веб-приложения

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

Задание 6. Создание базы данных -

© 2010-2023 Центр обучающих систем Сибирского федерального университета, sfu-kras.ru

Paspaбoтaнo на платформе moodle Beta-version (3.9.1.5.w3)

Политика конфиденциальности

Соглашение о Персональных данных

Политика допустимого использования

Контакты +7(391) 206-27-05 info-ms@sfu-kras.ru

Скачать мобильное приложение

Инструкции по работе в системе