

# Разработка программного обеспечения на языке Python

[Обзорная панель](#) ▶ [Мои курсы](#) ▶ [Разработка ПО на языке Python](#) ▶ [Веб-программирование на Python](#) ▶

[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 данных на сервер

```
<form method="POST" action="{% url 'create' %}">
  {% csrf_token %}
  <p><label>Введите имя</label><br>
  <input type="text" name="name" /></p>
```

00:00 / 04:31



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

- получение данных из бд
- добавление записи
- редактирование
- удаление

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

```

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

```

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 x
1  from django.shortcuts import render
2  from django.http import HttpResponse, HttpResponseRedirect
3  from .models import *
4
5
6  # получение данных из бд
7  def index(request):
8      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 x
13 # сохранение данных в бд
14 def create(request):
15     if request.method == "POST":
16         student = Student()
17         student.name = request.POST.get("name")
18         student.age = request.POST.get("age")
19
20         # получение id группы с формы
21         group_id = request.POST.get("group")
22
23         # получение записи связанной таблицы по id
24         group = Group.objects.get(id=group_id)
25
26         student.group = group
27         student.save()
28     return HttpResponseRedirect("/")

```

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.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8" />
5      <title>Деканат</title>
6  </head>
7  <body class="container">
8      <form method="POST" action="create/"...>
28
29      {% if students.count > 0 %}
30      <h2>Список студентов</h2>
31      <table>
32          <tr><th>Номер</th><th>Имя</th><th>Возраст</th><th>Группа</th></tr>
33          {% for student in students %}
34          <tr><td>{{ student.id }}</td><td>{{ student.name }}</td>
35          <td>{{ student.age }}</td><td>{{ student.group.name }}</td></tr>
36          {% endfor %}
37      </table>
38      {% endif %}
39  </body>
40  </html>

```

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.

```

<form method="POST" action="create/">
    {% csrf_token %}
    <p><label>Введите имя</label><br>
    <input type="text" name="name" /></p>
    <p><label>Введите возраст</label><br>
    <input type="number" name="age" /></p>
    <p><label>Выберите группу</label><br>
    <select name="group">
        {% for group in groups %}
        <option value="{{group.id}}">{{group.name}}</option>
        {% endfor %}
    </select>
    </p>
    <input type="submit" value="Добавить" >
</form>

```

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 `name` 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.

```
urls.py x
1 from django.urls import path
2 from .views import *
3
4 urlpatterns = [
5     path('', index),
6     path('create/', create),
7
8 ]
```

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.

Номер	Имя	Возраст	Группа
1	Ann	20	КИ21-11
2	Tom	23	КИ21-11
3		0	КИ21-11
4	Jake	25	КИ21-12

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).

```

forms.py
1  from django import forms
2      from django.forms import ModelForm
3  from .models import *
4
5
6  class StudentForm(ModelForm):
7      class Meta:
8          model = Student
9
10         fields = ['name', 'age', 'group']
11         labels = {
12             'name': 'Имя', 'age': 'Возраст', 'group': 'Группа'
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 %}
    <table>{{ form }}</table>
    <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

[◀ Задание 5. Добавление шаблонов страниц веб-приложения](#)[Перейти на...](#)

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

[Задание 6. Создание базы данных ▶](#)

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

Разработано 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](mailto:info-ms@sfu-kras.ru)

[Скачать мобильное приложение](#)[Инструкции по работе в системе](#)