Django

На настоящий момент последняя актуальная версия 6.6

- Сайт проекта https://www.djangoproject.com/
- Документация https://docs.djangoproject.com/en/6.0/
- Документация https://django.fun/docs/django/5.2/ (на русском)
- Документация https://metanit.com/python/django/ (на русском, но для версии 4.0)

Установка Django и создание первого проекта

Виртуальное окружение Python

Виртуальное окружение Python (`Python virtual environment`) – это инструмент, позволяющий создавать изолированные среды для выполнения и разработки приложений на языке Python.

С помощью виртуального окружения Python можно установить и использовать различные версии пакетов и зависимостей для каждого проекта, изолируя их друг от друга и предотвращая конфликты или несовместимости.

Используя виртуальное окружение Python, вы получаете:

1. Изоляцию зависимостей. Каждое виртуальное окружение имеет собственный независимый набор зависимостей и пакетов, что позволяет избежать конфликтов между различными версиями пакетов.
2. Управление версиями Python. Вы можете создавать и использовать различные версии Python в разных виртуальных окружениях, переключаясь между версиями, а также тестируя совместимость кода с новыми версиями Python.
3. Чистоту проекта. Виртуальное окружение помогает поддерживать проект организованным, так как все зависимости и пакеты проекта хранятся в отдельной директории. Это упрощает управление и развертывание проектов.
4. Переносимость. Вы можете передать виртуальное окружение на другую машину или другим разработчикам, что позволяет вести совместную работу над проектом.

Использование виртуальных окружений Python рекомендуется для разработки проектов, поскольку он помогает управлять и упрощать зависимости и версии, а также поддерживать проект в чистом и организованном состоянии.

Виртуальное окружение Python

Для создания виртуального окружения в Python рекомендуем использовать встроенный модуль venv. Для этого необходимо открыть командную строку и перейти в каталог, в котором будет создана виртуальная среда. Выполните команду:

```
python -m venv virt_name
```

где `virt_name` – имя вашей виртуальной среды. Дождитесь завершения процесса создания виртуальной среды. Это может занять некоторое время. Виртуальная среда будет создана в папке `virt_name`. В ней же будут находиться все файлы и зависимости, связанные с виртуальной средой.

Активация виртуального окружение с помощью команд:

- Для Windows: `.\virt_name\Scripts\activate`
- Для macOS и Linux: `source virt_name/bin/activate`

Установка Django

В терминале/консоли

Создали директория для проекта и зашли в нее

```
mkdir django
cd django
```

Установили и активировали виртуальную среду (назовем виртуальную среду .venv)

```
python -m venv .venv
.\.venv\Scripts\activate
```

Установили для данной виртуальной среды Django

```
python -m pip install Django
```

Создание нового проекта

В терминале/консоли

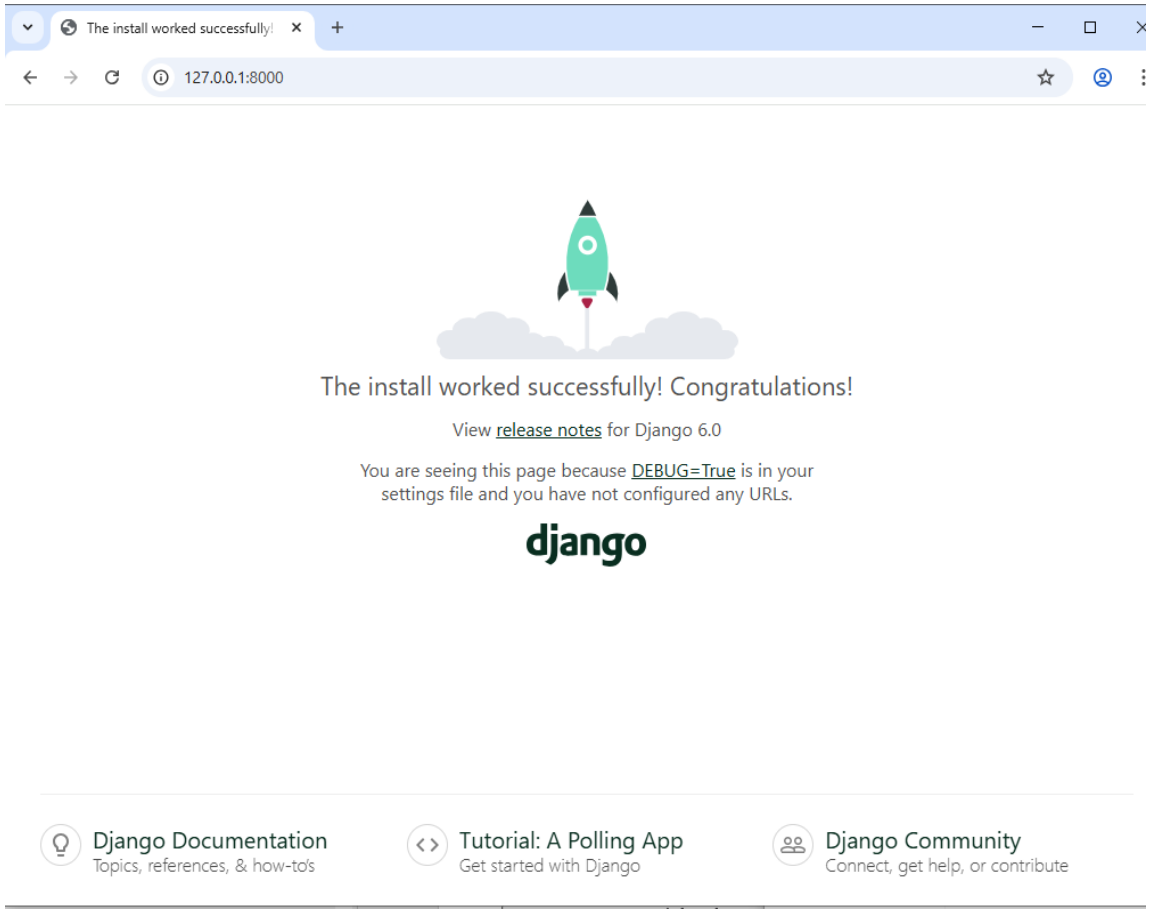
```
(.venv) $ django-admin startproject project
(.venv) $ tree /f project
|   manage.py
|
└───project
        asgi.py
        settings.py
        urls.py
        wsgi.py
        __init__.py
```

Запуск сервера

В терминале/консоли

```
(.venv) $ cd project
(.venv) $ python manage.py runserver
   Watching for file changes with StatReloader
   Performing system checks...
   System check identified no issues (0 silenced).
   You have 18 unapplied migration(s). Your project may
   not work properly until you apply the migrations for
   app(s): admin, auth, contenttypes, sessions.
   Run 'python manage.py migrate' to apply them.
   April 07, 2026 - 21:01:17
   Django version 6.0.4, using settings
   'new_project.settings'
   Starting development server at http://127.0.0.1:8000/
   Quit the server with CTRL-BREAK.
   WARNING: This is a development server. Do not use it in
   a production setting. Use a production WSGI or ASGI
   server instead.
   For more information on production servers see:
   https://docs.djangoproject.com/en/6.0/howto/deployment/
```

http://127.0.0.1:8000/



Ctrl-C останавливает сервер

Создание приложения в проекте

В терминале/консоли

```
(.venv) $ py manage.py startapp polls
(.venv) $ tree /f .
|   db.sqlite3
|   manage.py
|
+---polls
|   |   admin.py
|   |   apps.py
|   |   models.py
|   |   tests.py
|   |   views.py
|   |   __init__.py
|   |
|   \---migrations
|           __init__.py
|
\---project
        asgi.py
        settings.py
        urls.py
        wsgi.py
        __init__.py
```

`settings.py` База данных

project/settings.py

```
# Database
# https://docs.djangoproject.com/en/6.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

settings.py Локализация

project/settings.py

```
# Internationalization
# https://docs.djangoproject.com/en/6.0/topics/i18n/

LANGUAGE_CODE = 'ru'
TIME_ZONE = 'Europe/Moscow'
```

`settings.py` Installed Apps

project/settings.py

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin', # панель управления
    'django.contrib.auth', # система аутентификации
    'django.contrib.contenttypes', # система аутентификации
    'django.contrib.sessions', # библиотека для работы с сессиями пользователей
    'django.contrib.messages', # обработка сообщений
    'django.contrib.staticfiles ', # статических файлов
     'polls.apps.PollsConfig',
]
```

Чтобы включить приложение в проект, нам нужно добавить ссылку на его класс конфигурации, добавив в список INSTALLED_APPS элемент

`'polls.apps.PollsConfig',`

Миграции

Миграции – это способ Django распространять изменения, которые вносятся в модели (добавление поля, удаление модели и т.д.), в схему базы данных. Они разработаны, чтобы быть в основном автоматическими, но нужно знать, когда выполнять миграции, когда их запускать и с какими общими проблемами вы можете столкнуться.

Команды

Есть несколько команд, которые используются для взаимодействия с миграциями и обработкой Django схемы базы данных:

- `migrate`, которая отвечает за применение и отмену миграции.
- `makemigrations`, которая отвечает за создание новых миграций на основе изменений, которые вы внесли в свои модели.
- `sqlmigrate`, которая отображает операторы SQL для миграции.
- `showmigrations`, в которой перечислены миграции проекта и их статус.

Можно думать о миграции как о системе контроля версий для схемы базы данных. `makemigrations` отвечает за упаковку изменений модели в отдельные файлы миграции, а `migrate` отвечает за их применение в базе данных.

Файлы миграции для каждого приложения находятся в каталоге «`migrations`» внутри этого приложения и предназначены для передачи и распространения как часть его кодовой базы.

Запуск миграций

В терминале/консоли

```
(.venv) $ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, 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 auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Создание и активация модели

polls/models.py

```
from django.db import models

# Create your models here.
class Question(models.Model):
    question_text = models.CharField(verbose_name="Вопрос", max_length=200)
    pub_date = models.DateTimeField(verbose_name="Дата публикации")


class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE, verbose_name='Вопрос')
    choice_text = models.CharField(verbose_name="Ответ", max_length=200)
    votes = models.IntegerField(verbose_name="Количество голосов", default=0)
```

Данный фрагмент кода предоставляет Django множество информации. В частности,

- информация для создания необходимых структур в базе данных;
- методы, необходимые для удобной работы с объектами моделей.

Миграции

Создание файлов-миграций

```
(.venv) $ python manage.py makemigrations
Migrations for 'polls':
  polls\migrations\0001_initial.py
    + Create model Question
    + Create model Choice
```

Просмотр миграций

```
(.venv) $ python manage.py sqlmigrate polls 0001
BEGIN;
--
-- Create model Question
--
CREATE TABLE "polls_question" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "question_text" varchar(200) NOT NULL,
"pub_date" datetime NOT NULL);
--
-- Create model Choice
--
CREATE TABLE "polls_choice" ("id" integer NOT NULL PRIMARY KEY AUTOINCREMENT, "choice_text" varchar(200) NOT NULL,
"votes" integer NOT NULL, "question_id" bigint NOT NULL REFERENCES "polls_question" ("id") DEFERRABLE INITIALLY
DEFERRED);
CREATE INDEX "polls_choice_question_id_c5b4b260" ON "polls_choice" ("question_id");
COMMIT;
```

Запуск миграций

```
(.venv) $ python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, polls, sessions
Running migrations:
  Applying polls.0001_initial... OK
```

Консоль

```
(.venv) $ python manage.py shell
14 objects imported automatically (use -v 2 for details).
Ctrl click to launch VS Code Native REPL
Python 3.14.4 (tags/v3.14.4:23116f9, Apr  7 2026, 14:10:54) [MSC v.1944 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> Question.objects.all()
<QuerySet []>
>>> from django.utils import timezone
>>> timezone.now()
datetime.datetime(2026, 4, 7, 18, 50, 24, 978010, tzinfo=datetime.timezone.utc)
>>> q1 = Question(question_text="Что нового?", pub_date= timezone.now())
>>> q1
<Question: Question object (None)>
>>> q1.save()
>>> q1
<Question: Question object (1)>
>>> q2 = Question(question_text="Что нового еще?", pub_date=timezone.now())
>>> q2
<Question: Question object (None)>
>>> q2.save()
>>> q2
<Question: Question object (2)>
>>> Question.objects.all()
<QuerySet [<Question: Question object (1)>, <Question: Question object (2)>]>
>>> Question.objects.get(pk=2)
<Question: Question object (2)>
>>> Question.objects.get(pk=3)
Traceback (most recent call last):
  ...
polls.models.Question.DoesNotExist: Question matching query does not exist.
>>> exit()
now exiting InteractiveConsole...
```

Консоль

Для упрощения работы с консолью (сохранение истории и т.д.):

1) установим:

```
(.venv) $ python  -m pip install django-extensions ipython
```

2) добавим `'django_extensions'` в `INSTALLED_APPS` в файле `settings.py`:

```
INSTALLED_APPS = [
        ...
        'django_extensions',
]
```

3) запустим

```
(.venv) python manage.py shell_plus
```

Перезагрузка модулей при изменениях

```
# после запуска shell_plus
load_ext autoreload
# по мере надобности
autoreload
```

Модифицируем модели

polls/models.py

```
from django.db import models
from django.utils import timezone
import datetime


class Question(models.Model):
    question_text = models.CharField(verbose_name="Вопрос", max_length=200)
    pub_date = models.DateTimeField(verbose_name="Дата публикации")


    def __str__(self):
        return f"Вопрос: {self.question_text}"
    def was_published_recently(self):
        return self.pub_date >= timezone.now()-datetime.timedelta(days=1)


class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE, verbose_name='Вопрос')
    choice_text = models.CharField(verbose_name="Ответ", max_length=200)
    votes = models.IntegerField(verbose_name="Количество голосов", default=0)


    def __str__(self):
        return f"Ответ {self.choice_text}, ответили {self.votes}"
```

Консоль

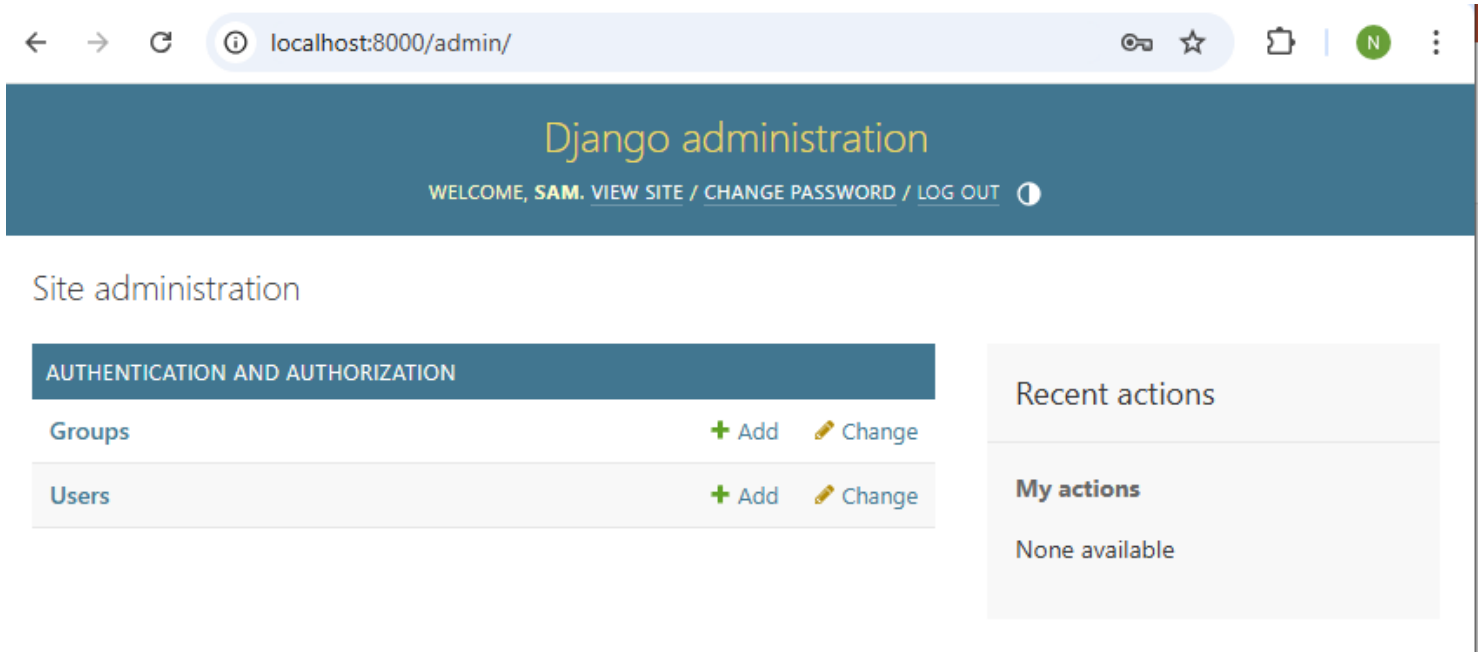
```
(venv) $ python manage.py shell_plus
In [1]: Question.objects.all()
Out[1]: <QuerySet [<Question: Вопрос: Что нового?>, <Question: Вопрос: Что нового?>]>
In [2]: Question.objects.first().was_published_recently()
Out[2]: True
In [3]: q = Question.objects.first()
In [4]: Choice.objects.create(question=q, choice_text = "Ничего", votes = 2)
Out[4]: <Choice: Ответ Ничего, ответили 2>
In [5]: Choice.objects.all()
Out[5]: <QuerySet [<Choice: Ответ Ничего, ответили 2>]>
In [6]: Choice.objects.first().question
Out[6]: <Question: Вопрос: Что нового?>
In [7]: q.choice_set.all()
Out[7]: <QuerySet [<Choice: Ответ Ничего, ответили 2>]>
In [8]: q.choice_set.create(choice_text='Немного', votes=0)
Out[8]: <Choice: Ответ Немного, ответили 0>
In [9]: q.choice_set.create(choice_text='Много всего')
Out[9]: <Choice: Ответ Много всего, ответили 0>
In [10]: q.choice_set.create(choice_text='Не скажу')
Out[10]: <Choice: Ответ Не скажу, ответили 0>
In [11]: q.choice_set.all()
Out[11]: <QuerySet [<Choice: Ответ Ничего, ответили 2>, <Choice: Ответ Немного, ответили 0>, <Choice: Ответ Много всего, ответили 0>, <Choice:
Ответ Не скажу, ответили 0>]>
In [12]: q.choice_set.filter(choice_text='Не скажу')
Out[12]: <QuerySet [<Choice: Ответ Не скажу, ответили 0>]>
In [13]: q.choice_set.filter(choice_text__startswith='Не')
Out[13]: <QuerySet [<Choice: Ответ Немного, ответили 0>, <Choice: Ответ Не скажу, ответили 0>]>
In [14]: c = q.choice_set.filter(choice_text__startswith='Не')
In [15]: c.delete()
Out[15]: (2, {'polls.Choice': 2})
In [16]: q.choice_set.all()
Out[16]: <QuerySet [<Choice: Ответ Ничего, ответили 2>, <Choice: Ответ Много всего, ответили 0>]>>>>
```

Панель управления

Создание администратора

```
(.venv) $ py manage.py createsuperuser
Username (leave blank to use 'natalie'): sam
Email address: sam@email.com
Password:
Password (again):
Superuser created successfully.
(.venv) $ py manage.py runserver
```

http://localhost:8000/admin



Добавление моделей в панель управления

polls/admin.py

```
from django.contrib import admin
from .models import Question, Choice


admin.site.register(Question)
admin.site.register(Choice)
```

Дополнительная настройка панели управления

polls/admin.py

```
from django.contrib import admin
from .models import Question, Choice

class QuestionAdmin(admin.ModelAdmin):
    list_display = ('question_text', 'pub_date')
    list_display_links = ('question_text', 'pub_date')
    search_fields = ('question_text',)
admin.site.register(Question, QuestionAdmin )

class ChoiceAdmin(admin.ModelAdmin):
    list_display = ('question', 'choice_text', 'votes')
    list_display_links = ('question', 'choice_text', 'votes')
    search_fields = ('choice_text',)
admin.site.register(Choice, ChoiceAdmin)
```

- **list_display** – последовательность имен полей, которые должны выводиться в списке записей;
- **list_display_links** – последовательность имен полей, которые должны быть преобразованы в гиперссылки, ведущие на страницу правки записей;
- **search_fields** – последовательность имен полей, по которым должна выполняться фильтрация.

https://docs.djangoproject.com/en/6.0/ref/contrib/admin/

Метаданные модели

```python
class Question(models.Model):
    question_text = models.CharField(verbose_name="Вопрос", max_length=200)
    pub_date = models.DateTimeField(verbose_name="Дата публикации")


    def __str__(self):
        return f"Вопрос: {self.question_text}"
    def was_published_recently(self):
        return self.pub_date >= timezone.now()-datetime.timedelta(days=1)

    class Meta:
        verbose_name = 'Вопрос'
        verbose_name_plural = 'Вопросы'
        ordering = ["-pub_date"]


class Choice(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE, verbose_name='Вопрос')
    choice_text = models.CharField(verbose_name="Ответ", max_length=200)
    votes = models.IntegerField(verbose_name="Количество голосов", default=0)


    def __str__(self):
        return f"Ответ {self.choice_text}, ответили {self.votes}"

    class Meta:
        verbose_name = 'Ответ'
        verbose_name_plural = 'Ответы'
        ordering = ["choice_text"]
```

https://docs.djangoproject.com/en/6.0/ref/models/options/#available-meta-options