



Фреймворк Django

разработка веб-приложений

For beginners



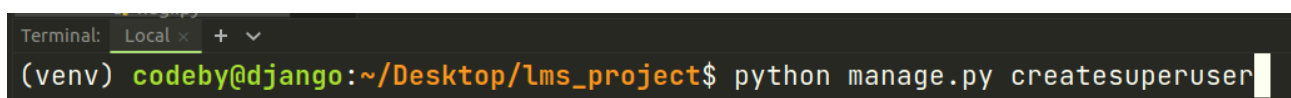
Урок 4. Административная панель

Создание суперпользователя

Административный сайт Django предоставляет полный набор функционала для наполнения, редактирования и удаления данных из БД и имеет интуитивно понятный интерфейс, с которым мы познакомимся совсем скоро.

Доступ к административной панели может получить пользователь, являющийся суперпользователем (в этом случае поля `is_superuser` и `is_staff` таблицы, хранящей пользовательские данные, принимают значение `True`) или пользователь со статусом персонала (т. е. в пользовательской таблице только в поле `is_staff` лежит значение `True`).

Для создания суперпользователя нам необходимо воспользоваться командой утилиты `manage.py`: `python manage.py createsuperuser`



```
Terminal: Local x + v
(venv) codeby@django:~/Desktop/lms_project$ python manage.py createsuperuser
```

После нажатия клавиши `Enter` в терминале Вам предложат ввести идентификационные данные для успешной регистрации: адрес электронной почты и пароль, который нужно придумать самостоятельно. В процессе ввода пароль не будет отображаться на экране в целях безопасности. Вам необходимо будет ввести его 2 раза.



```
Terminal: Local x + v
(venv) codeby@django:~/Desktop/lms_project/lms_project$ python manage.py createsuperuser
Email: admin@example.com
Password:
Password (again):
```

В том случае, если Вы введёте слишком простой пароль, то Django предупредит Вас об этом и предложит снова ввести пароль, но уже надёжней.

```
(venv) codeby@django:~/Desktop/lms_project/lms_project$ python manage.py createsuperuser
Email: admin@example.com
Password:
Password (again):
The password is too similar to the Email.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]:
```

Как видим, Django сравнивает пароль на предмет схожести с адресом электронной почты, проверяет его на длину (не менее 8 символов) и проверяет его на распространённость.


Если Вы хотите всё-таки придумать более сложный пароль, то в ответ на вопрос терминала введите букву «y», в противном случае «N».

После ввода пароля Django инициировал следующую ошибку и соответственно первая попытка создания суперпользователя провалилась:

```
Terminal: Local x + v
return super().execute(*args, **options)
File "/home/codeby/Desktop/lms_project/venv/lib/python3.8/site-packages/django/core/ma
output = self.handle(*args, **options)
File "/home/codeby/Desktop/lms_project/venv/lib/python3.8/site-packages/django/contrib
195, in handle
    self.UserModel._default_manager.db_manager(database).create_superuser(**user_data)
TypeError: create_superuser() missing 1 required positional argument: 'username'
(venv) codeby@django:~/Desktop/lms_project/lms_project$
```

Давайте разберёмся, что представляет из себя эта ошибка?

Всё очень просто! Помните, в прошлом уроке мы при создании пользовательской модели объявляли атрибут **REQUIRED_FIELDS** и присвоили ему пустой список.

```
13  REQUIRED_FIELDS = []
```

Напомню, что этот атрибут отвечает за обязательные поля, которые необходимо заполнить при создании суперпользователя. Так как мы решили немного облегчить себе путь создания модели **User** (без создания менеджера по

управлению процессом создания различных пользователей), то в данном случае применяются инструкции по умолчанию. И они требуют, чтобы при создании суперпользователя обязательно заполнялось поле **username**. Поэтому параметр **REQUIRED_FIELDS** необходимо скорректировать, добавив в него значение в виде названия данного поля.

```
13 REQUIRED_FIELDS = ['username']
```

Теперь повторим очередную попытку создания суперпользователя, введя всё ту же команду:

```
Terminal: Local x + v
(venv) codeby@django:~/Desktop/lms_project/lms_project$ python manage.py createsuperuser
Email: admin@example.com
Username: admin
Password:
Password (again):
```

Мы уже видим, что при вводе, в отличие от прошлой попытки, Django стал требовать поле **username**.

После ввода корректного пароля Django выведет сообщение об успешности создания суперпользователя.

```
Terminal: Local x + v
(venv) codeby@django:~/Desktop/lms_project/lms_project$ python manage.py createsuperuser
Email: admin@example.com
Username: admin
Password:
Password (again):
Superuser created successfully.
(venv) codeby@django:~/Desktop/lms_project/lms_project$
```

Заглянем в БД и проверим корректность создания записи суперпользователя в таблице **auth_app_user**.

auth_app_user x +							
<div> <div>↺ 1 ↻</div> <div>🔍</div> <div>⚙️ Insert rows</div> <div>📄 Export</div> <div>0.006s Results: 1 lms_project</div> </div>							
id	password	last_login	is_superuser	username	first_name	last_name	
1	pbkdf2_sha256\$320000\$kcswtWhG213DXLGOQgs...	NULL	1	admin			
is_staff	is_active	date_joined	email	birthday	description	avatar	
1	1	2022-07-24 13:21:15.204758	admin@example.com	NULL			

Как можно увидеть, заполнены только поля: **password** (в виде хеша — для безопасного хранения паролей), **is_superuser** принял значение True, **admin** с значением из терминала, которое мы ранее ввели, **is_staff** и **is_active** также в значение True, в поле **date_joined** внесена дата с временной меткой, отражающая точное время создания пользователя, **email** с значением из терминала. Поле **last_login** нужно для хранения времени последнего входа пользователя на сайт. Поскольку мы ещё не выполнили ни одного входа, то оно имеет значение NULL.

На данный момент этого недостаточно, чтобы начать знакомство с интерфейсом админки Django. Также необходимо:

- зарегистрировать в параметре **INSTALLED_APPS** следующие приложения: **django.contrib.admin**, **django.contrib.auth**, **django.contrib.contenttypes**, **django.contrib.sessions** и **django.contrib.messages**.

Что это за приложения было разъяснено ранее в уроке, посвящённом описанию файла настроек проекта.

```

31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     # Custom apps
41     'auth_app.apps.AuthAppConfig',
42     'learning.apps.LearningConfig',
43 ]

```

- зарегистрировать в параметре **MIDDLEWARE** следующие обработчики:
 django.contrib.sessions.middleware.SessionMiddleware,
 django.contrib.auth.middleware.AuthenticationMiddleware,
 django.contrib.messages.middleware.MessageMiddleware.

```

45 MIDDLEWARE = [
46     'django.middleware.security.SecurityMiddleware',
47     'django.contrib.sessions.middleware.SessionMiddleware',
48     'django.middleware.common.CommonMiddleware',
49     'django.middleware.csrf.CsrfViewMiddleware',
50     'django.contrib.auth.middleware.AuthenticationMiddleware',
51     'django.contrib.messages.middleware.MessageMiddleware',
52     'django.middleware.clickjacking.XFrameOptionsMiddleware',
53 ]

```

- зарегистрировать во вложенном параметре **context_processors** ключа **OPTIONS** следующие обработчики контекста для шаблонизатора:
 django.contrib.auth.context_processors.auth,
 django.contrib.messages.context_processors.messages.


```

57  TEMPLATES = [
58      {
59          'BACKEND': 'django.template.backends.django.DjangoTemplates',
60          'DIRS': [],
61          'APP_DIRS': True,
62          'OPTIONS': {
63              'context_processors': [
64                  'django.template.context_processors.debug',
65                  'django.template.context_processors.request',
66                  'django.contrib.auth.context_processors.auth',
67                  'django.contrib.messages.context_processors.messages',
68              ],
69          },
70      ],
71  ]

```

- добавить к списку маршрутов `urlpatterns` в файле `urls.py`, в директории с настройками проекта, маршрут для взаимодействия непосредственно с административной панелью. Все маршруты для успешной работы с админкой Django уже предоставляет. Для этого нужно импортировать модуль `admin` из `django.contrib`. А затем обратиться к атрибуту `urls` объекта, представляющего админку, переменной `site`.

```

16  from django.contrib import admin
17  from django.urls import path
18
19  urlpatterns = [
20      path('admin/', admin.site.urls),
21  ]

```

- провести миграции (это мы сделали уже в прошлом уроке при создании моделей)
- создать суперпользователя (это мы сделали недавно)

Эти минимальные условия нужно выполнить для успешного запуска административного сайта.

К счастью, все эти параметры были созданы при создании проекта и нам пришлось лишь проверить их наличие.

Знакомство с интерфейсом административной панели

Для начала работы с админкой нам нужно запустить сервер командой: `python manage.py runserver`

```
Terminal: Local x + v

System check identified no issues (0 silenced).
July 24, 2022 - 16:02:41
Django version 4.0, using settings 'lms_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

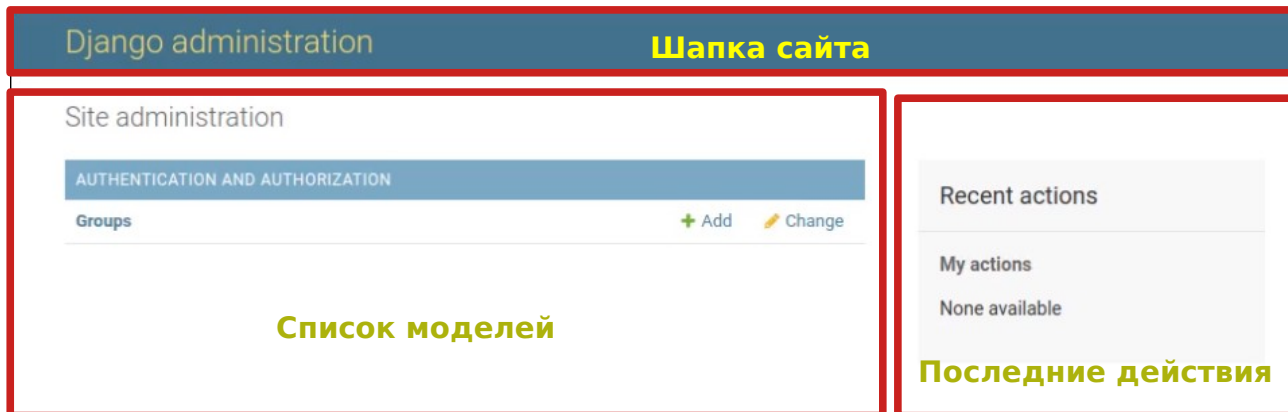
Скопируем подсвеченную IDE ссылку и вставим её в адресную строку браузера, прибавив к ней ранее назначенный путь к админке `admin/`.



Перейдя по этой ссылке мы увидим обычную форму для входа на сайт.

Введём в соответствующие поля ранее созданные **email** и **password**, затем нажмём на кнопку «Log in».

После нажатия мы увидим главную страницу админки сайта:



Основная страница содержит следующие блоки:

- шапка с названием сайта (позже мы её поменяем)
- в левой части мы можем найти все зарегистрированные модели. По умолчанию в админке зарегистрирована модель **Group**, служащая для хранения разрешений групп пользователей.
- в правой части хранятся последние выполненные действия



В правой части шапки сайта отображается **email** пользователя, под которым был выполнен вход, ссылка на главную страницу сайта, ссылка, позволяющая выйти с сайта, и ссылка для изменения пароля текущего пользователя, перейдя по которой мы увидим следующую страницу с формой:

Django administration

WELCOME, ADMIN@EXAMPLE.COM. [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Password change

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

«

Password change

Please enter your old password, for security's sake, and then enter your new password twice so we can verify you typed it in correctly.

Old password:

New password:

Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.

New password confirmation:

[CHANGE MY PASSWORD](#)

Также после входа мы можем заметить, что в поле таблицы `auth_app_user` в поле `last_login` появилось значение времени входа в виде всё той же даты с временной меткой:

auth_group			
auth_app_user			
1			
Insert rows			
Export			
Id			
password			
last_login			
1	pbkdf2_sha256\$320000\$sk...	2022-07-24 16:03:16.871559	

Также мы можем видеть, что контент сайта отображается на английском языке, что не совсем будет удобно для понимания инструкций.

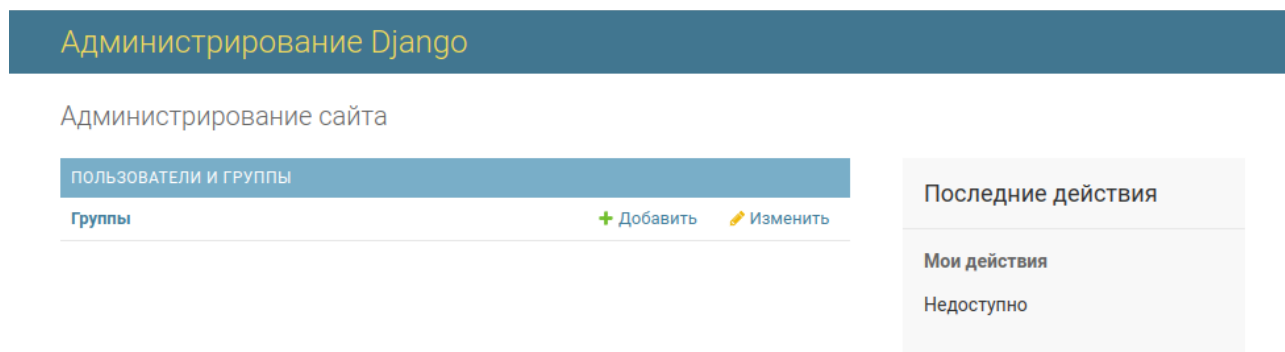
Для этого перейдём в файл настроек проекта `settings.py` и поменяем значение параметра `LANGUAGE_CODE` с „en“ на „ru“.

```
109 # Internationalization
110 # https://docs.djangoproject.com/en/4.0/topics/i18n/
111
112 LANGUAGE_CODE = 'ru'
```

После этих изменений сайта сервер автоматически перезапустится, так как обнаружит изменения в проекте (это мы можем наблюдать в терминале IDE). После этого мы можем обновить страницу админки и увидеть, что контент перевёлся на русский язык.

Коды для всех языков Вы можете найти на сайте:

https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes



Регистрация моделей в административной панели

Для привязки конкретной модели к админке Django для дальнейшего упрощения манипулирования данными в таблицах необходимо всё тем же образом обратиться к методу **register** объекта административного сайта **admin**

Но перед этим нам необходимо импортировать нужные модели для регистрации. Зарегистрируем модели сначала для приложения **auth_app**.

Данное приложение содержит всего одну модель **User**, поэтому её и импортируем

```
3 from .models import User
```

И далее, проведём регистрацию

```
5 # Register your models here.  
6 admin.site.register(User)
```

Также пока уберём с админки модель **Group**, её рассмотрение отложим на отдельный урок, связанный с разграничением доступа. Добиться этого можно, применив противоположный метод — **unregister**. Но сначала её необходимо импортировать из модуля **django.contrib.auth.models**.

```
2 from django.contrib.auth.models import Group
7 admin.site.unregister(Group)
```

Перейдём в админку и увидим, что модель **User** действительно появилась, в свою очередь модель **Group** исчезла.

Администрирование Django

Администрирование сайта

AUTH_APP

Участники

+ Добавить

✎ Изменить

Мы видим, что таблица приняла имя, указанное в прошлом уроке в классе **Meta** в параметре **verbose_name_plural**.

```
15 class Meta:
16     verbose_name_plural = 'Участники'
```

Можно заметить, что название приложения, в котором находится модель осталось не адаптированным к языковым настройкам. Изменим это, применив параметр **verbose_name** в файле конфигурации приложения **apps.py**

```
4 class AuthAppConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'auth_app'
7     verbose_name = 'Управление авторизацией'
```

Перезапустим сервер, обновим страницу и зафиксируем изменения:

Администрирование Django

Администрирование сайта

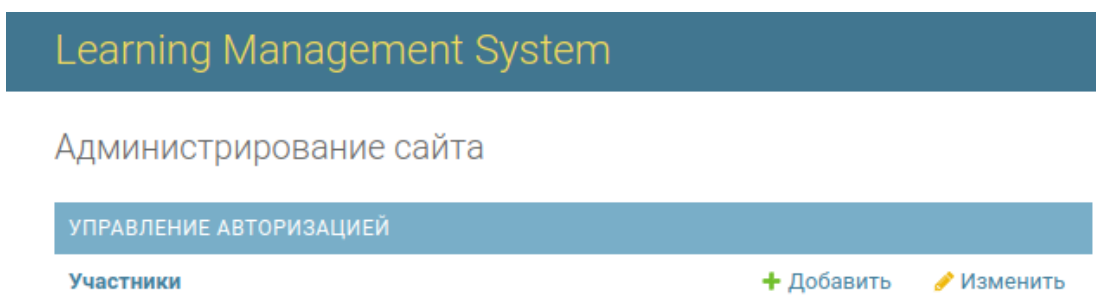


Также поменяем default-название шапки админки.

Сделать это можно, поменяв параметр `site_header` объекта админки.

```
8 admin.site.site_header = 'Learning Management System'
```

Теперь шапка выглядит так, название успешно поменялось:



Теперь давайте изучим, что предоставляет админка для манипулирования данными.

В самом начале справа от названия модели есть 2 кнопки для добавления / изменения записи модели



Поле названия модели является ссылкой, по которой мы можем перейти для просмотра всех записей модели.

Выберите Участник для изменения

[ДОБАВИТЬ УЧАСТНИК +](#)

Действие: ▼ [Выполнить](#) Выбрано 0 объектов из 1

- ☐ УЧАСТНИК
- ☐ Участник : admin@example.com

1 Участник

На данный момент мы видим ранее созданного суперпользователя. Его запись представляет собой строку, переопределённую в методе `__str__` модели `User`.

```
20 def __str__(self):  
21     return f'Участник {self.first_name} {self.last_name}: {self.email}'
```



Также внизу мы видим агрегированное значение в виде общего количества записей в модели. Позже мы отредактируем вывод под себя. Перейдём по ссылке на него. У нас появится форма со всеми полями таблицы для редактирования записи:

Изменить Участник

[ИСТОРИЯ](#)

Участник : admin@example.com

Пароль:

Последний вход:
Дата: [Сегодня](#) | 
Время: [Сейчас](#) | 

Внимание: Ваше локальное время опережает время сервера на 3 часа.

☒ Статус суперпользователя
Указывает, что пользователь имеет все права без явного их назначения.

Группы:

Группы, к которым принадлежит данный пользователь. Пользователь получит все права, указанные в каждой из его/её групп. Удерживайте "Control" (или "Command" на Mac), чтобы выбрать несколько значений.

Нажав кнопку «История» мы сможем посмотреть всю историю изменений конкретной записи.

История изменений: Участник : admin@example.com

ДАТА И ВРЕМЯ	ПОЛЬЗОВАТЕЛЬ	ДЕЙСТВИЕ
24 июля 2022 г. 19:43	admin@example.com	Изменено Дата рождения.
24 июля 2022 г. 19:43	admin@example.com	Ни одно поле не изменено.
24 июля 2022 г. 19:43	admin@example.com	Ни одно поле не изменено.

В конце формы имеются 3 кнопки, подразумевающие различный порядок действий после сохранения записи.

Сохранить и добавить другой объект

Сохранить и продолжить редактирование

СОХРАНИТЬ

Кнопка «Сохранить и добавить другой объект» говорит нам о том, что после сохранения текущей записи Django автоматически перенаправит нас на страницу с пустой формой для создания новой записи.

Кнопка «Сохранить и продолжить редактирование» говорит нам о том, что после сохранения текущей записи не будет никакого редиректа и Вы сможете по своему желанию продолжать редактировать ту же самую запись.

Кнопка «Сохранить» говорит нам о том, что после сохранения текущей записи Django перенаправит нас на страницу со всеми записями модели.

Удалить

Кнопка «Удалить» не делает ничего необычного, кроме удаления записи и редиректа на страницу с полным списком записей текущей модели.

Теперь перейдём к регистрации моделей приложения **learning**.

Для этого мы будем использовать немного другой способ, а именно мы привяжем модели с помощью наследника класса `ModelAdmin`. Он предоставляет более расширенные возможности для кастомизации представления модели в админке сайта, а именно:

- настройка списка выводимых полей модели
- настройка списка полей для поиска
- настройка списка полей, которые станут гиперссылками для перехода на страницу правки записи

Импортируем классы моделей `Course` и `Lesson`, создадим классы-наследники `ModelAdmin` и затем применим декоратор для регистрации в админке сайта.

```
1  from django.contrib import admin
2  from .models import Course, Lesson
3
4
5  @admin.register(Course)
6  class CourseAdmin(admin.ModelAdmin):
7      pass
8
9
10 @admin.register(Lesson)
11 class LessonAdmin(admin.ModelAdmin):
12     pass
```

Оставим эти классы без реализации, так как сначала нам нужно убедиться в корректности отображения в админке.

Перезапустим сервер, обновим страницу и увидим, что данные модели успешно появились в админке сайта.

Администрирование сайта

LEARNING		
Курсы	+ Добавить	✎ Изменить
Уроки	+ Добавить	✎ Изменить

Так же, как и в приложении `auth_app`, изменим название приложения, добавив атрибут `verbose_name` в конфигурационный класс `LearningConfig`.

```
4 class LearningConfig(AppConfig):
5     default_auto_field = 'django.db.models.BigAutoField'
6     name = 'learning'
7     verbose_name = 'Управление обучением'
```

Зафиксируем полученные изменения.

УПРАВЛЕНИЕ ОБУЧЕНИЕМ		
Курсы	+ Добавить	✎ Изменить
Уроки	+ Добавить	✎ Изменить

Далее нам придётся произвести процесс заполнения таблиц данными для наглядного представления применённых настроек отображения записей модели.

Пример заполнения модели `Course`

Выберите Курс для изменения

Действие: Выбрано 0 объектов из 5

☐ курс

☐ Django Framework: Старт 2022-07-06

☐ HTML-вёрстка: Старт 2022-09-01

☐ JavaScript: Старт 2022-08-19

☐ Python Language: Старт 2022-09-23

☐ Веб-дизайн: Старт 2022-07-31

5 Курсы

Пример заполнения модели Lesson

Также настроим отображение названия записей, переопределив метод `__str__` в модели Lesson.

```
33 def __str__(self):
34     return f'{self.course.title}: Урок {self.name}'
```

И в итоге получим следующее:

Выберите Урок для изменения

Действие: Выбрано 0 объектов из 6

- ☐ УРОК
- ☐ Django Framework: Урок Контроллеры-классы
- ☐ Django Framework: Урок Контроллеры-функции
- ☐ Django Framework: Урок Маршрутизация
- ☐ Django Framework: Урок Административная панель
- ☐ Django Framework: Урок Структура проекта
- ☐ Django Framework: Урок Введение

6 Уроки

Пришло время познакомиться с атрибутами, которые предоставляет класс `ModelAdmin`.

- `list_display` — позволяет указать список выводимых полей

```
5 @admin.register(Course)
6 class CourseAdmin(admin.ModelAdmin):
7     list_display = ('title', 'author', 'start_date', )
```

Выберите Курс для изменения

ДОБАВИТЬ КУРС +

Действие: Выбрано 0 объектов из 5

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА	СТАРТ КУРСА
<input type="checkbox"/>	Django Framework	Участник : admin@example.com	6 июля 2022 г.
<input type="checkbox"/>	HTML-вёрстка	Участник : admin@example.com	1 сентября 2022 г.
<input type="checkbox"/>	JavaScript	Участник : admin@example.com	19 августа 2022 г.
<input type="checkbox"/>	Python Language	Участник : admin@example.com	23 сентября 2022 г.
<input type="checkbox"/>	Веб-дизайн	Участник : admin@example.com	31 июля 2022 г.

5 Курсы

- **exclude** — напротив, позволяет указать список полей, которые НЕ нужно выводить. При этом все остальные поля будут выведены. Следует указывать только один из этих атрибутов, т. е. либо список выводимых или список НЕвыводимых полей, в противном случае возникнет ошибка.

```
5 @admin.register(Course)
6 class CourseAdmin(admin.ModelAdmin):
7     # list_display = ('title', 'author', 'start_date', )
8     exclude = ('description', 'duration', 'price', )
```

Изменить Курс

Django Framework: Старт 2022-07-06

Название курса:	<input type="text" value="Django Framework"/>
Автор курса:	<input type="text" value="Участник : admin@example.com"/> <input type="button" value="✎"/> <input type="button" value="✚"/> <input type="button" value="✖"/>
Старт курса:	<input type="text" value="06.07.2022"/> Сегодня <input type="button" value="📅"/> <small>Внимание: Ваше локальное время опережает время сервера на 3 часа.</small>
Кол-во уроков:	<input type="text" value="20"/>

Удалить

- `search_fields` — позволяет указать перечень полей, по которым будет производиться поиск.

```
8 search_fields = ('title', 'start_date', 'description', )
```

Выберите Курс для изменения

Выберите Курс для изменения

ДОБАВИТЬ КУРС +

1 результат (5 всего)

Действие: Выбрано 0 объектов из 1

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА	СТАРТ КУРСА
<input type="checkbox"/>	HTML-вёрстка	Участник : admin@example.com	1 сентября 2022 г.

1 Курс

Также он позволяет указать следующие префиксы перед названием поля:

- «`=`» - точное совпадение при поиске по данному полю, т. е. текст в форме для поиска должен полностью совпадать со значением в БД. Регистр символов не учитывается.

```
8 search_fields = ('=title', )
```

Если мы введём «веб», ничего не выведется, но при значении «веб-дизайн» появится соответствующий курс.

0 результатов (5 всего)

1 результат (5 всего)

Действие: Выбрано 0 объектов из 1

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА	СТАРТ КУРСА
<input type="checkbox"/>	Веб-дизайн	Участник : admin@example.com	31 июля 2022 г.

1 Курс

- «[^]» - слово в форме обязательно должно присутствовать в начале данного поля.

```
8 search_fields = ('^title', )
```

Q language Найти 0 результатов (5 всего)

Q python Найти 1 результат (5 всего)

Действие: ----- Выполнить Выбрано 0 объектов из 1

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА	СТАРТ КУРСА
<input type="checkbox"/>	Python Language	Участник : admin@example.com	23 сентября 2022 г.

1 Курс

- «@» - нужен для полнотекстового поиска (поддерживается только MySQL).

Данный класс позволяет настроить порядок вывода записей модели с помощью следующих атрибутов:

- `list_per_page` — число записей на одной странице в виде числа типа `int`.

```
9 list_per_page = 3
```

Выберите Курс для изменения

Q Найти

Действие: ----- Выполнить Выбрано 0 объектов из 3

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА
<input type="checkbox"/>	Django Framework	Участник : admin@example.com
<input type="checkbox"/>	HTML-вёрстка	Участник : admin@example.com
<input type="checkbox"/>	JavaScript	Участник : admin@example.com

1 2 5 Курсы Показать все

- `actions_on[top|bottom]` — логический атрибут, если True, выводит все доступные действия для выбранных записей

```
10  actions_on_top = True  
11  actions_on_bottom = True
```

Выберите Курс для изменения

Действие:

Выбрано 2 из 3

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА
<input checked="" type="checkbox"/>	Django Framework	Участник : admin@example.com
<input type="checkbox"/>	HTML-вёрстка	Участник : admin@example.com
<input checked="" type="checkbox"/>	JavaScript	Участник : admin@example.com

Действие:

Выбрано 0 объектов из 3

1 2 5 Курсы [Показать все](#)

- `action_selection_counter` — логический атрибут, если True, выведет количество выбранных записей около списка с множественным выбором.

```
12   actions_selection_counter = True
```

Действие:

Выбрано 2 из 3

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА
<input type="checkbox"/>	Django Framework	Участник : admin@example.com
<input checked="" type="checkbox"/>	HTML-вёрстка	Участник : admin@example.com
<input checked="" type="checkbox"/>	JavaScript	Участник : admin@example.com

- `save_on_top` — позволяет добавить блок с кнопками для управления записью сверху формы для редактирования

Изменить Курс

ИСТОРИЯ

Django Framework: Старт 2022-07-06

Удалить

Сохранить и добавить другой объект

Сохранить и продолжить редактирование

СОХРАНИТЬ

Название курса:

Django Framework

Автор курса:

Участник : admin@example.com

✎ + ✖

- `list_display_links` — перечень полей, которые станут гиперссылками на форму для правки записи

```
list_display_links = ('title', 'start_date', )
```

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА	СТАРТ КУРСА	ОПИСАНИЕ КУРСА
<input type="checkbox"/>	Django Framework	Участник : admin@example.com	6 июля 2022 г.	Разработка серверных веб-приложений
<input type="checkbox"/>	HTML-вёрстка	Участник : admin@example.com	1 сентября 2022 г.	Изучение основ вёрстки веб-страниц, основных тегов
<input type="checkbox"/>	JavaScript	Участник : admin@example.com	19 августа 2022 г.	Изучение языка программирования для придания динамики вашему сайту

- `list_editable` — перечень полей, редактирование которых будет производиться непосредственно в таблице с записями модели, т. е. без перенаправления на страницу с формой, содержащей полный список полей

<input type="checkbox"/>	НАЗВАНИЕ КУРСА	АВТОР КУРСА	СТАРТ КУРСА	ОПИСАНИЕ КУРСА
<input type="checkbox"/>	Django Framework	Участник : admin@example.com	6 июля 2022 г.	Разработка серверных веб-приложений
<input type="checkbox"/>	HTML-вёрстка	Участник : admin@example.com	1 сентября 2022 г.	Изучение основ вёрстки веб-страниц, основных тегов
<input type="checkbox"/>	JavaScript	Участник : admin@example.com	19 августа 2022 г.	Изучение языка программирования для придания динамики вашему сайту

Действие: Выполнить Выбрано 0 объектов из 3

1 2 5 Курсы Показать все

Сохранить

Пусть полем для редактирования станет `description`. В этом случае все элементы в этой ячейке станут обычными тегами `textarea` в терминах HTML.

```
<textarea name="form-0-description" cols="40" rows="10" class="vLargeTextField"
maxlength="200" id="id_form-0-description" style="height: 25px; width: 619px;">
Разработка серверных веб-приложений</textarea>
```

Для сохранения изменений нам потребуются лишь нажать в правом нижнем углу кнопку «Сохранить».

Для модели **Lesson** применим аналогичные атрибуты:

```
15 @admin.register(Lesson)
16 class LessonAdmin(admin.ModelAdmin):
17     list_display = ('course', 'name', 'preview', )
18     search_fields = ('name', )
19     list_per_page = 3
20     actions_on_top = False
21     actions_on_bottom = True
22     actions_selection_counter = True
```

Выберите Урок для изменения

ДОБАВИТЬ УРОК +

<input type="checkbox"/>	КУРС	НАЗВАНИЕ УРОКА	ОПИСАНИЕ УРОКА
<input type="checkbox"/>	Django Framework: Старт 2022-07-06	Контроллеры-классы	Изучение основных классов для отображения данных модели
<input type="checkbox"/>	Django Framework: Старт 2022-07-06	Контроллеры-функции	Научиться обрабатывать входящие запросы с помощью функций
<input type="checkbox"/>	Django Framework: Старт 2022-07-06	Маршрутизация	Варианты подключения маршрутов к приложению

Действие: Выбрано 0 объектов из 3

1 2 6 Уроки [Показать все](#)

На настоящий момент ограничимся данными понятиями административной панели Django. В процессе расширения нашего проекта мы будем кастомизировать представление моделей и узнаем новые способы отображения данных.

Домашнее задание

- ✓ Создать суперпользователя и подключить административную панель к проекту
- ✓ Поменять язык, контент шапки админки
- ✓ Сделать привязку своих моделей к админке сайта
- ✓ Кастомизировать представление данных, применив соответствующие атрибуты