

Python для продвинутых специалистов (Django)



Тема №10. Работа с данными через Административную панель

Преподаватель: Панченко Игорь
Валентинович

Административная панель Django

Что такое административная панель

Фреймворк Django имеет встроенную административную панель, которая позволяет решить множество прикладных задач:

- Создавать и редактировать модели
- Распределять права доступа для групп пользователей
- Формировать интерфейс для взаимодействия пользователей с моделями и многое другое

Базовые настройки

Для первичного входа в админ-панель, необходимо создать аккаунт супер-пользователя. Это делается командой:

`python manage.py createsuperuser`

Для изменения языка административной панели, вы можете изменить значение переменной **`LANGUAGE_CODE`** на **`'ru-RU'`** в файле **`settings.py`**

Регистрация моделей

Для того, чтобы модели отображались в панели администратора, их необходимо регистрировать. В файлах **admin.py** соответствующих приложений, например:

```
from .models import *
class ArticleAdmin(admin.ModelAdmin):
    list_filter = ['title', 'date', 'author']
    list_display = ['title', 'date', 'author']

admin.site.register(Article, ArticleAdmin)
```

Импортируем модель, создаём соответствующий класс Tag, добавляем настройки, регистрируем модель специальной командой

Настройка параметров отображения

В результате зарегистрированные модели появляются в панели администратора:

NEWS

Tags + Add ✎ Change

Новости + Add ✎ Change

Если зайти в конкретную модель, там будут отображаться поля и фильтры, настроенные на предыдущем шаге:

Select Новость to change

Action:

▼

Go

0 of 3 selected

| <input type="checkbox"/> | НАЗВАНИЕ | ▲ | ДАТА ПУБЛИКАЦИИ | AUTHOR |
|--------------------------|---------------|---|--------------------------|--------|
| <input type="checkbox"/> | Новость IT #1 | | Nov. 19, 2023, 7:54 p.m. | Admin |

Заголовок страницы админ.панели

Вы можете настроить заголовок страницы админ.панели, добавив в главный **Urls.py** файл следующую команду:

```
admin.site.site_header = "Панель администрирования"
```

Панель администрирования

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

 Add  Change

Users

 Add  Change

Заголовок страницы админ.панели

Чтобы изменить следующий заголовок,
добавьте команду:

**admin.site.index_title = “Новости нашего
города”**

Панель администрирования

Новости нашего города

AUTHENTICATION AND AUTHORIZATION

Groups

 Add  Change

Users

 Add  Change

Имена моделей

Чтобы дать моделям приятные глазу имена, в **Meta-данных** классов необходимо определить переменные **verbose_name, verbose_name_plural:**

```
#метаданные класса Article
class Meta:
    ordering = ['title']
    verbose_name = 'Новость'
    verbose_name_plural = 'Новости'
```

Имя таблицы панели администратора

Чтобы изменить имя таблицы в админ-панели, нужно добавить атрибут `verbose_name` в файле **ИмяПриложения/apps.py**:

```
class NewsConfig(AppConfig):  
    default_auto_field = 'django.db.models.BigAutoField'  
    name = 'news'  
    verbose_name = 'Новости'
```

НОВОСТИ

Tags

+ Add

 Change

Новости

+ Add

 Change

Полезные функции файла `admin.py`

По умолчанию, только левое поле является ссылкой на конкретную запись в таблице БД:

| <input type="checkbox"/> | НАЗВАНИЕ | ДАТА ПУБЛИКАЦИИ | AUTHOR |
|--------------------------|---------------|--------------------------|--------|
| <input type="checkbox"/> | Новость IT #1 | Nov. 19, 2023, 7:54 p.m. | Admin |

Если определить переменную `list_display_links`, вы можете сделать несколько полей ссылками:

```
class ArticleAdmin(admin.ModelAdmin):  
    list_filter = ['title', 'date', 'author']  
    list_display = ['title', 'date', 'author']  
    list_display_links = ('title', 'author')
```

| <input type="checkbox"/> | НАЗВАНИЕ | ДАТА ПУБЛИКАЦИИ | AUTHOR |
|--------------------------|---------------|--------------------------|--------|
| <input type="checkbox"/> | Новость IT #1 | Nov. 19, 2023, 7:54 p.m. | Admin |

Полезные функции файла **admin.py**

Вы можете изменить порядок сортировки моделей при отображении в панели администратора, на порядок запроса данных из БД это не повлияет. Для этого в админ-классе добавьте атрибут **ordering**:

```
class ArticleAdmin(admin.ModelAdmin):  
    ordering = ['date', 'title', 'author']
```

Чтобы сортировать в обратном порядке – добавьте знак минус.

Регистрация через декоратор

Кстати, вы можете регистрировать модели не только при помощи команды

admin.site.register(), но и при помощи специального декоратора: **@admin.register:**

```
@admin.register(Tag)
class TagAdmin(admin.ModelAdmin):
    list_filter = ['title', 'status']
    list_display = ['title', 'status']
```

Он выполняет аналогичную функцию

Механизмы работы с полями

Редактируемые поля списка

Вы можете настроить возможность редактирования некоторых полей в списочном отображении:

Action: 0 of 3 selected

| <input type="checkbox"/> | НАЗВАНИЕ | 2 ▲ | ДАТА ПУБЛИКАЦИИ | 1 ▲ | АВТОР | 3 ▲ |
|--------------------------|---|-----|---------------------------|-----|----------|-----|
| <input type="checkbox"/> | <input type="text" value="Новость IT #1"/> | | Nov. 19, 2023, 7:54 p.m. | | Admin | |
| <input type="checkbox"/> | <input type="text" value="Новость IT #2"/> | | Nov. 19, 2023, 7:54 p.m. | | Admin | |
| <input type="checkbox"/> | <input type="text" value="Новость медицины"/> | | Nov. 21, 2023, 11:37 p.m. | | DemoUser | |

Пользователю не придётся заходить в отдельные записи для их редактирования

Редактируемые поля списка

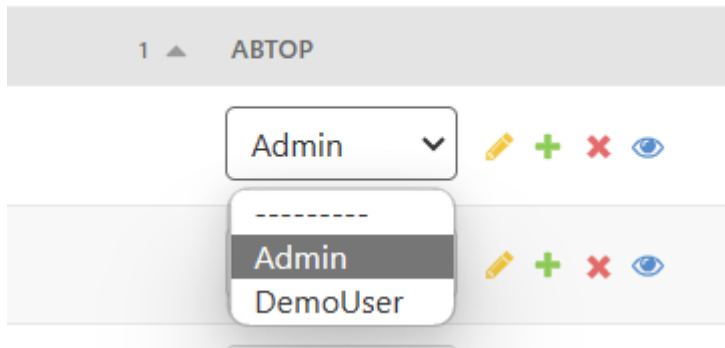
Для этого, необходимо задать список таких полей в атрибуте **list_editable**:

```
class ArticleAdmin(admin.ModelAdmin):  
    list_editable = ['title']
```

Важно! Поля, которые вы хотите редактировать не должны быть включены в набор **list_display_links** (по-умолчанию, это нулевое поле)

Редактируемые поля списка

Вы можете даже редактировать поля со связями один-к-одному и один-ко-многим:



Но, к сожалению M2M редактировать таким образом не получится

Поля только для чтения

Вы можете защитить поля от редактирования, перечислив их в наборе **readonly_fields**:

```
readonly_fields = ('date',)
```

Доступ к изменению значений этих полей будет заблокирован из панели администратора. Таким образом – вы можете защитить данные пользователей от редактирования недобросовестными администраторами

Предзаполненные поля

Вы можете создавать предзаполненные поля, которые берут содержимое из других полей модели. Часто таким образом создаётся поле slug для ссылок, например:

```
class ArticleAdmin(admin.ModelAdmin):  
    prepopulated_fields = {"slug": ("title",)}
```

Поле Slug будет заполняться содержимым из поля title при вводе символов в **title**

Больше различных полей

Познакомьтесь с документацией, чтобы иметь представление о всех возможностях админ-панели:

[The Django admin site | Django documentation | Django \(djangoproject.com\)](#)

Пагинация

Вы можете управлять пагинацией внутри панели администратора, для этого вы можете настроить значение атрибута:

```
list_per_page = 2
```

До:

| <input type="checkbox"/> | НАЗВАНИЕ | 2 |
|--------------------------|------------------|---|
| <input type="checkbox"/> | Новость IT №1 | |
| <input type="checkbox"/> | Новость IT №2 | |
| <input type="checkbox"/> | Новость медицины | |

После:

| <input type="checkbox"/> | НАЗВАНИЕ | 2 | ДАН |
|-----------------------------|---------------|---|-----|
| <input type="checkbox"/> | Новость IT №1 | | Nov |
| <input type="checkbox"/> | Новость IT №2 | | Nov |
| 1 2 3 Новости Show all Save | | | |

Пользовательские поля в админ-панели

Пользовательские поля

В панели администратора вы можете показывать не только поля, присущие модели, но и самодельные поля. Они могут содержать:

- Краткое описание модели (количество символов в новости, количество использований тега и т.д.)
- Удобные подписи/подсказки
- Связанные с моделями изображения
- И многое другое

**Пользовательское поле – количество
символов в статье**

Подсчет символов в статье

Часто для определения длины статьи используются 2 параметра: количество символов, либо число слов. Создадим пользовательское поле, отображающее данную информацию. Это можно сделать либо внутри модели, либо прямо в админ-классе

```
list_per_page = 2









def symbols_count(self, article: Article):
    return f"Количество символов: {len(article.text)}"
```

Подсчет символов в статье

Для отображения этого значения в админ-панели, добавляем его в **list_display**:

```
list_display = ['title', 'date', 'author', 'symbols_count']
```

Результат:

| <input type="checkbox"/> | НАЗВАНИЕ | 2 ▲ | ДАТА ПУБЛИКАЦИИ | 1 ▲ | АВТОР | 3 ▲ | SYMBOLS COUNT |
|--------------------------|---------------|-----|--------------------------|-----|---------|---|-------------------------|
| <input type="checkbox"/> | Новость IT №1 | | Nov. 19, 2023, 7:54 p.m. | | Admin ▼ |     | Количество символов: 13 |
| <input type="checkbox"/> | Новость IT №2 | | Nov. 19, 2023, 7:54 p.m. | | Admin ▼ |     | Количество символов: 46 |

Изменение названия пользовательского поля

Чтобы изменить название поля, используйте декоратор `@admin.display`:

```
@admin.display(description='Длина')  
def symbols_count(self, article: Article):  
    return f"Количество символов: {len(article.text)}
```

Название поля изменилось, но мы не можем производить по нему сортировку. Django не понимает, по какому критерию сортировать эту колонку

| ▲ | длина |
|---|-------------------------|
| | Количество символов: 13 |
| | Количество символов: 46 |

Сортировка по пользовательскому полю

К сожалению, мы можем только связать сортировку пользовательского поля с уже существующим полем описанным в модели, например, сортировать по полю text (Даже, если мы не отображаем его в админ-панели):

```
@admin.display(description='Длина', ordering='text')
def symbols_count(self, article: Article):
    return f"Количество символов: {len(article.text)}"
```

▲ ДЛИНА 1 ▼

Количество символов: 16

Количество символов: 46

Сортировка по длине статьи

Чтобы реализовать сортировку по пользовательскому полю – необходимо переопределить метод **get_queryset** внутри админ-класса, которым пользуется встроенный механизм сортировки

Сортировка по пользовательскому полю

Мы обращаемся к родительскому методу `Get_queryset`, аннотируем поле `'text'` нашей статьи функцией `Length` и возвращаем аннотированный `queryset` с дополнительным полем `_symbols`:

```
from django.db.models.functions import Length

def get_queryset(self, request):
    queryset = super().get_queryset(request)
    queryset = queryset.annotate(_symbols=Length('text'))
    return queryset
```

Поправляем декоратор

Добавляем новое поле в параметре ordering

```
@admin.display(description='Длина', ordering='_symbols')
def symbols_count_display(self, article: Article):
    return f"Количество символов: {len(article.text)}"
```

Результат:

| ДЛИНА | 1 |
|-------------------------|---|
| Количество символов: 46 | |
| Количество символов: 16 | |
| Количество символов: 13 | |

Количество использований тега

Самостоятельная работа

По аналогии с полем `symbols_count`, добавьте в админ-панель количество использований Tag. Используйте агрегационную функцию **Count**

Решение на следующем слайде

| <input type="checkbox"/> | TITLE | 2 ▲ | STATUS | 3 ▲ | ИСПОЛЬЗОВАНИЙ | 1 ▼ |
|--------------------------|----------|-----|--------|-----|---------------|-----|
| <input type="checkbox"/> | IT | | ✓ | | 2 | |
| <input type="checkbox"/> | Medicine | | ✓ | | 1 | |

Решение

```
@admin.register(Tag)  
class TagAdmin(admin.ModelAdmin):  
    list_filter = ['title', 'status']  
    list_display = ['title', 'status', 'tag_count']  
@admin.display(description='Использований', ordering='tag_count')  
    def tag_count(self, obj):  
        return obj.tag_count  
  
    def get_queryset(self, request):  
        queryset = super().get_queryset(request)  
        queryset = queryset.annotate(tag_count=Count("article"))  
        return queryset
```

Отображение связанных изображений

Добавляем поле для хранения изображений

Добавим в нашу базу данных новый класс Images, с помощью которого мы будем добавлять изображения, привязанные к НОВОСТЯМ:

```
class Image(models.Model):
    article = models.ForeignKey(Article, on_delete=models.CASCADE)
    title = models.CharField(max_length=50, blank=True)
    image = models.ImageField(upload_to='article_images/')

    def __str__(self):
        return self.title

    def image_tag(self):
        return mark_safe(f'')
```

Добавляем поле для хранения изображений

Выполните миграцию. Зарегистрируйте **Image**
В **admin.py**:

```
class ImageAdmin(admin.ModelAdmin):  
    list_filter = ['title', 'id']  
    list_display = ['title', 'id', 'image_tag']  
  
admin.site.register(Image, ImageAdmin)
```


Внутри класса **Image** определен метод **Image_tag**, который формирует ссылку на изображение

Image_tag

При помощи функции `mark_safe`, мы помечаем кусок HTML-кода как безопасный для визуализации в шаблоне. В результате, мы увидим не просто текстовое значение, а отрендеренное в шаблоне изображение:

Select image to change

Action: 0 of 1 selected

| <input type="checkbox"/> | TITLE | ID | IMAGE TAG |
|--------------------------|-----------|----|---|
| <input type="checkbox"/> | Диаграмма | 1 |  |

Отображение изображений в привязке к новостям










Так как изображения связаны с Article через связь один-ко-многим, нам сперва необходимо запросить изображения из БД, затем выбрать одно из них, и сформировать для него ссылку. Добавьте метод в класс **Article**:

```
def image_tag(self):  
    image = Image.objects.filter(article=self)  
    if image:  
        return mark_safe(f'')  
    else:  
        return '(No image)'
```

Отображение изображений в привязке к новостям

Не забудьте добавить **image_tag** в **list_display** админ-класса **ArticleAdmin**.

Результат:

| | | | | | | | | |
|--------------------------|------------------------------------|-----------------------------------|-----------------|---|-------------------------|--|--|--|
| Action: | <input type="text" value="-----"/> | <input type="button" value="Go"/> | 0 of 2 selected | | | | | |
| <input type="checkbox"/> | НАЗВАНИЕ 2 ▲ | ДАТА ПУБЛИКАЦИИ 1 ▲ | АВТОР 3 ▲ | ДЛИНА | IMAGE TAG | | | |
| <input type="checkbox"/> | Новость IT №1 | Nov. 19, 2023, 7:54 p.m. | Admin ▼ |     | Количество символов: 13 |  | | |
| <input type="checkbox"/> | Новость IT №2 | Nov. 19, 2023, 7:54 p.m. | Admin ▼ |     | Количество символов: 46 | (No image) | | |

**Встроенное редактирование
связанных записей**

Встроенное редактирование связанных записей

Часто у пользователей возникает необходимость добавлять связанные записи при создании основных: например, вы создаёте новый Article и хотите сразу загрузить необходимые изображения. Для того, на данный момент вам придётся создать Article, перейти в другой раздел Admin-панели, что не очень удобно.

Для решения этой проблемы, есть специальные инструменты, называемые **Inlines**

ImagesInline

Реализуем описанный выше пример. Для этого, внутри файла **admin.py**, мы определяем новый класс:

```
class ArticleImageInline(admin.TabularInline):  
    model = Image  
    extra = 3  
    readonly_fields = ('id', 'image_tag')
```

Связанная модель – Image, **Extra** – количество полей для изображений

ImagesInline

Добавляем атрибут `inlines` к админ-классу:

```
class ArticleAdmin(admin.ModelAdmin):
    list_editable = ['author']
    ordering = ['date', 'title', 'author']
    list_filter = ['title', 'date', 'author']
    list_display = ['title', 'date', 'author', 'symbols_count_display', 'image_tag']
    list_display_links = ('title',)
    readonly_fields = ('date',)
    list_per_page = 2
    inlines = (ArticleImageInline,)
```

Результат

Теперь мы видим все изображения, связанные с конкретной новостью. Так же – мы можем редактировать поля и удалять эти файлы.

| IMAGES | | | | |
|--------------------------------|--|--|--------------------------|--|
| TITLE | IMAGE | IMAGE TAG | DELETE? | |
| Диаграмма | <div>Currently: article_images/form_handling_-_standard.png</div> <div>Change: <div>Выбор файла</div> Не выбран ни один файл</div> |  | <input type="checkbox"/> | |
| <div></div> | <div>Выбор файла</div> Не выбран ни один файл | - | <input type="checkbox"/> | |
| <div></div> | <div>Выбор файла</div> Не выбран ни один файл | - | <input type="checkbox"/> | |
| <div></div> | <div>Выбор файла</div> Не выбран ни один файл | - | <input type="checkbox"/> | |
| <div>+ Add another Image</div> | | | | |

Действия в панели администратора

Действия в панели администратора

При работе с моделями в панели администратора доступны действия:

Select image to change

Action: 0 of 1 selected

| <input type="checkbox"/> | TITLE | ID |
|--------------------------|------------------------|----|
| <input type="checkbox"/> | ----- | |
| | Delete selected images | |

По-умолчанию, вы можете удалять выбранные записи из какой-либо таблицы. Также – вы можете создавать пользовательские действия

Действия – set_status

Создадим действия для тегов – установление значения поля **status** в **True/False**. Описываем метод `set_true`, и добавляем его в список `actions` внутри класса `TagAdmin`

```
class TagAdmin(admin.ModelAdmin):  
    list_filter = ['title', 'status']  
    list_display = ['title', 'status', 'tag_count']  
    actions = ['set_true']  
  
    @admin.action(description="Активировать выбранные теги")  
    def set_true(self, request, queryset):  
        count = queryset.update(status=True)  
        self.message_user(request, f'Активировано {count} тег(ов)')
```


Действия – set_status

Наше действие получает queryset с выбранными тегами, считает их количество, обновляет значение полей status у указанных тегов, и отправляет сообщение об успешности операции

```
class TagAdmin(admin.ModelAdmin):  
    list_filter = ['title', 'status']  
    list_display = ['title', 'status', 'tag_count']  
    actions = ['set_true']  
  
    @admin.action(description="Активировать выбранные теги")  
    def set_true(self, request, queryset):  
        count = queryset.update(status=True)  
        self.message_user(request, f'Активировано {count} тег(ов)')
```

Самостоятельная работа

Создайте своё действие **set_false** для деактивации выбранных тегов.

Придумайте действие для модели Article и попробуйте его реализовать.

Панель поиска и фильтрация

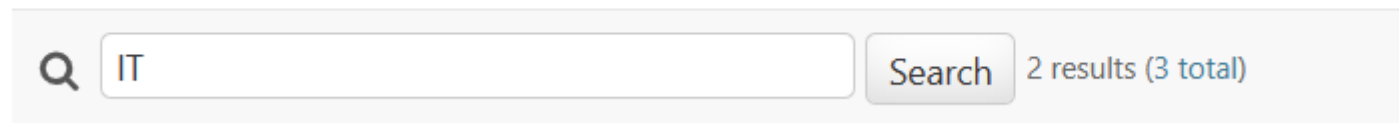
Панель поиска

Когда записей становится слишком много, в них становится трудно ориентироваться. В таком случае становится целесообразной настройка панели поиска.

Чтобы появилась панель поиска, в админ-классе необходимо определить атрибут:

```
search_fields = ['title']
```

Select Новость to change

A screenshot of a search bar interface. It features a magnifying glass icon on the left, followed by a text input field containing the text "IT". To the right of the input field is a button labeled "Search". Further to the right, the text "2 results (3 total)" is displayed in a smaller font.

Q IT Search 2 results (3 total)

Поиск по тегам





Добавим возможность поиска по тегам. Для этого, нам необходимо добавить поиск по полю **title** наших объектов **Tag**:

```
search_fields = ['title', 'tags__title']
```

Select Новость to change

1 result (3 total)

Action: 0 of 1 selected

| <input type="checkbox"/> | НАЗВАНИЕ 2 ▲ | ДАТА ПУБЛИКАЦИИ 1 ▲ | АВТОР 3 ▲ | ДЛИНА | IMAGE TAG |
|--------------------------|---------------------|---------------------------|--|----------------------------|---------------|
| <input type="checkbox"/> | Новость медицины | Nov. 21, 2023, 11:37 p.m. | DemoUser ▼     | Количество символов: 16 | (No image) |

Поиск по lookup-ам

В поисковых полях вы можете использовать знакомые вам lookup-функции (contains, exact и т.д.), например, чтобы искомый фрагмент тайтла искался с начала заголовка:

```
search_fields = ['title__startswith', 'tags__title']
```

Таким образом вы можете настраивать панель поиска

Настройка панели фильтрации

Мы уже знаем, что для фильтрации по значениям полей достаточно определить атрибут **list_filter** внутри админ-класса:

```
list_filter = ['title', 'date', 'author']
```

Так же, вы можете создавать пользовательские фильтры для ваших моделей

Пользовательский фильтр

```
class ArticleFilter(admin.SimpleListFilter):
    title = 'Фильтр по длине новости'
    parameter_name = 'text'

    def lookups(self, request, model_admin):
        return [
            ('S', ("Короткие, <100 зн.")),
            ('M', ("Средние, 100-500 зн.")),
            ('L', ("Длинные, >500 зн.")),
        ]

    def queryset(self, request, queryset):
        if self.value() == "S":
            return queryset.annotate(text_len=Length('text')).filter(text_len__lt=100)
        elif self.value() == "M":
            return queryset.annotate(text_len=Length('text')).filter(text_len__lt=500,
                                                                       text_len__gte=100)
        else:
            return queryset.annotate(text_len=Length('text')).filter(text_len__gte=500)
```


Пользовательский фильтр

Title - название фильтра в админ-панели.
Parameter_name – название параметра в поисковом запросе (GET). Метод Lookups возвращает список значений параметра фильтрации, queryset() отвечает за отбор записей для соответствия выбранному варианту фильтрации

↓ Ву Фильтр по длине новости"

All

Короткие, <100 зн.

Средние, 100-500 зн.

Длинные, >500 зн.

Настройка формы редактирования записей

Настройка формы редактирования записей

По-умолчанию, Django отображает все поля модели в форме редактирования:

Change Тэг

Crypto

HISTORY

Title:

Crypto

☒ Status

SAVE

Save and add another

Save and continue editing

Delete

Однако, и этот список можно настроить, определив список `fields` внутри админ-класса

Список отображаемых полей

Например, настроим поля для Article:

```
class ArticleAdmin(admin.ModelAdmin):  
    fields=['title', 'text']  
    """ ... """
```

Change Новость

Новость: **Длинная новость.** Дата создания: 2023-11-28 22:18:56.643305+00:00

HISTORY

VIEW ON SITE >

Название:

Длинная новость

Текст новости:

Длинная новость Длинная новость Длинная новость Длинная новость Длинная новость Длинная

Как видите, появились только 2 поля, при этом, в порядке перечисления полей в переменной **fields**

Список исключаемых полей

Либо вы можете пойти с другой стороны и указать список исключаемых полей **exclude**:

```
class ArticleAdmin(admin.ModelAdmin):  
    # fields=['title', 'text' ]  
    exclude = ['author', 'title']
```

Change Новость

Новость: Длинная новость. Дата создания: 2023-11-28 22:18:56.643305+00:00

HISTORY

VIEW ON SITE >

Tags:

Crypto
Economics
IT
Medicine
Politics
Science

+

Hold down "Control", or "Command" on a Mac, to select more than one.

Аннотация:

Длинная новость

Автозаполняемое поле slug

Что такое slug

Slug – Это поле, которое часто используют, чтобы формировать уникальную url-ссылку, не используя поля PK или ID (например, чтобы не показывать конкурентам, сколько всего у вас в магазине товаров, сколько всего на сайте новостных статей и т.д.). Добавим это поле для наших статей и выполним миграции:

```
class Article(models.Model):  
    #поля  
    slug = models.SlugField(default="", null=True, blank=True)
```

Автозаполнение поля

Новость: Длинная новость. Дата создания: 2023-11-28 22:18:56.643305+00:00

Slug:

Далее, заполним наш slug текстом из поля text:

```
class ArticleAdmin(admin.ModelAdmin):  
    prepopulated_fields = {"slug": ("title",)}
```

Мы указываем из какого поля нужно брать информацию для формирования slug.

slug

Результат выглядит так:

Новость: **Длинная новость.** Дата создания: 2023-11-28 22:18:56.643305+00:00

Slug:

dlinnaya-novost

Название:

Длинная новость

Теперь, когда мы меняем значение поля “title”, slug меняется автоматически. Как видите, Django автоматически умеет писать транслитом с русского языка

Фильтры для M2M полей

Фильтры для M2M полей

При работе с полями M2M вы можете добавить горизонтальный или вертикальный фильтры. Это делается добавлением атрибута `filter_horizontal` или `filter_Vertical` внутри админ-класса.

Это бывает полезно, когда у вас много связанных объектов, например – десятки существующих тегов.

filter_vertical=['tags']

Tags:

Available tags ?

Q

Filter

Crypto
Economics
IT
Medicine
Politics
Science

+

↓ ↑

Chosen tags ?

Q

Filter

Hold down "Control", or "Command" on a Mac, to select more than one.

filter_vertical['tags']

Tags:

Available tags ?

Q Filter

Crypto
Economics
IT
Medicine
Politics
Science

Choose all ?

Chosen tags ?

Q Filter

Remove all

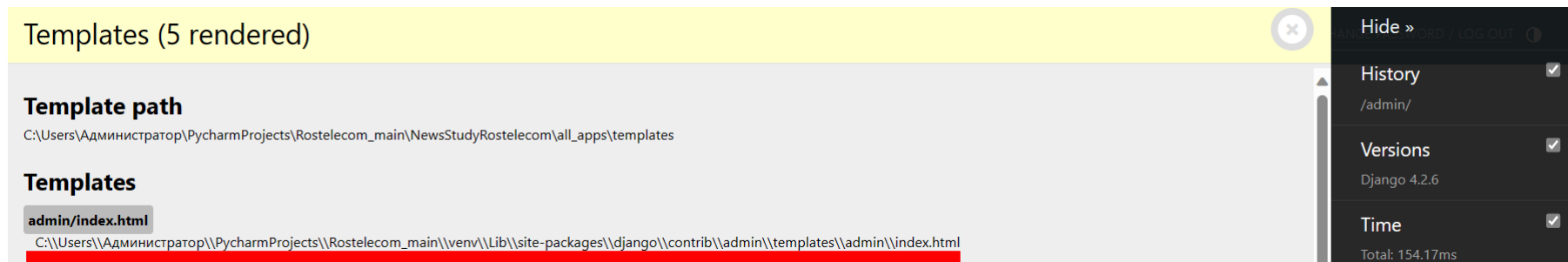
Внешний вид админ-панели

Внешний вид админ-панели

Часто, возникает необходимость стилизовать админ-панель под стиль основного сайта.

Django Даёт некоторые возможности в этом направлении.

Если вы откроете Debug Toolbar, вы сможете увидеть какие шаблоны используются для формирования страницы админ-панели:



The screenshot displays the Django Debug Toolbar, specifically the 'Templates' panel. The panel title is 'Templates (5 rendered)'. It shows the 'Template path' as 'C:\Users\Администратор\PycharmProjects\Rostelecom_main\NewsStudyRostelecom\all_apps\templates'. Under the 'Templates' section, the template 'admin/index.html' is highlighted, with its path shown as 'C:\Users\Администратор\PycharmProjects\Rostelecom_main\venv\Lib\site-packages\django\contrib\admin\templates\admin\index.html'. On the right side of the toolbar, a sidebar contains links for 'History', 'Versions', and 'Time', each with a checkbox and a 'Hide »' button.

| Template path |
|---|
| C:\Users\Администратор\PycharmProjects\Rostelecom_main\NewsStudyRostelecom\all_apps\templates |

| Templates |
|---|
| admin/index.html |
| C:\Users\Администратор\PycharmProjects\Rostelecom_main\venv\Lib\site-packages\django\contrib\admin\templates\admin\index.html |

| History | Versions | Time |
|---------|--------------|-----------------|
| /admin/ | Django 4.2.6 | Total: 154.17ms |

Шаблоны админ-панели

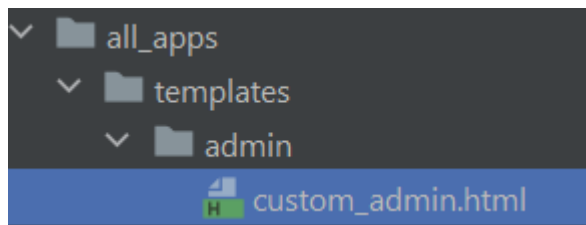
Templates

- [admin/index.html](#) ...\\Rostelecom_main\\venv\\Lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\index.htmlToggle
- [admin/base_site.html](#) ...\\Rostelecom_main\\venv\\Lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\base_site.htmlToggle
- [admin/base.html](#) ...\\Rostelecom_main\\venv\\Lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\base.htmlToggle
- [admin/color_theme_toggle.html](#) ...\\Rostelecom_main\\venv\\Lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\color_theme_toggle.htmlToggle
- [admin/app_list.html](#) ...\\Rostelecom_main\\venv\\Lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\app_list.html

Шаблоны админ-панели

Базовым для страниц сайта является шаблон **base_site.html**.

Создайте в общем каталоге шаблонов, доступном для всех приложений подкаталог **admin** и внутри него шаблон **custom_admin**:



В главном файле **urls** укажите:

```
admin.site.index_template = 'admin/custom_admin.html'
```

Шаблоны админ-панели

Перенесите содержимое из стандартного шаблона **base_site.html** в **custom_admin.html**

Мы сделали этот манёвр, чтобы не изменять исходные файлы админ-шаблонов, зато теперь мы можем свободно редактировать **custom_admin**

Элементы базового шаблона

{{ **title** }} – заголовок вкладки в браузере

{{ **branding** }} – отвечает за надпись в верхней части страницы



Панель администрирования

Home > Новости нашего города

{{ **nav-global** }} отвечает за навигационную панель в правом верхнем углу:



WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)



Пользовательские стили

В файле `base_site.html` нет ссылок на стили, значит они скрыты в шаблоне верхнего уровня, от которого расширяется данный файл:

```
{% extends "admin/base.html" %}
```

В нём мы обнаруживаем:

```
{% block extrastyle %}{% endblock %}
```

Это как раз тот блок, который служит для назначения дополнительных стилей оформления

Пользовательские стили

Подключим в этот файл наш файл стилей:

```
2  {% load static %}
3  {% block extrastyle %}
4      <link rel="stylesheet" href="{% static 'main/css/style.css' %}">
5  {% endblock %}
```

Теперь мы можем присваивать стили элементам по их ID, например:

```
<div id="header">
<div id="branding">
```

```
/*стили админ-панели: */
#header {
    background: #212529;
    color: white; /* цвет текста */
}
```

```
div.breadcrumbs {
    background: #485159;
}
.module caption {
    background: #485159;
    color: #000000;
}
```

Результат

Панель администрирования

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

DJDT

Home > Новости нашего города

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

HOME

Demos [+ Add](#)

USERS

Accounts [+ Add](#)

НОВОСТИ

Images [+ Add](#)

Новости [+ Add](#)

Тэги [+ Add](#)

Новости нашего города

Резюме

Вы так же можете переопределить исходный шаблон от которого наследуется шаблон админ-панели и назначить там своё содержимое и свои классы элементов. Таким образом, вы можете создать админ-панель обладающую уникальным интерфейсом, но работающую со всеми механизмами админ-панели Django.

Материалы

[Как сделать свою страницу в Django Admin с выразительной Hand Chart? / Хабр \(habr.com\)](#)

[сортировка в списке администратора django по пользовательскому полю в алфавитном порядке | Все о фреймворке Джанго и его библиотеках](#)

[Руководство Django часть 4: административная панель Django - Изучение веб-разработки | MDN \(mozilla.org\)](#)

[Getting Started - django-filter 23.4 documentation](#)