Django: админ-зона | Я.Шпора

Админ-зона доступна сразу после старта сервера Django:
http://127.0.0.1:8000/admin/ .

Создание суперпользователя

Суперпользователь создаётся командой `createsuperuser`:

```
python manage.py createsuperuser
```

Система запросит имя для пользователя, адрес его почты и пароль. Под этой учётной записью можно войти в админку и управлять проектом.

Добавление моделей в админ-панель

Чтобы модели отображались в админке — модели надо зарегистрировать в файлах *<имя_приложения>/admin.py*:

```
# <имя_приложения>/admin.py
from django.contrib import admin

# Из модуля models импортируем модель Category...
from .models import Category

# ...и регистрируем её в админке:
admin.site.register(Category)
```

Локализация и перевод

Чтобы переключить интерфейс админки на другой язык, надо найти кодовое обозначение нужного языка и подставить его в константу `LANGUAGE_CODE` в файле *settings.py*:

```
# <project_dir>/settings.py

# Переводим на русский!
LANGUAGE_CODE = 'ru-RU'
```

Человекочитаемые названия: verbose_name

Атрибут `verbose_name` позволяет изменять:

Название приложения: создайте класс-наследник класса `AppConfig` в файле `apps.py`. Назовите класс по имени приложения, в котором находится файл `apps.py`. Добавьте в класс атрибут `verbose_name`:

```
# <имя_приложения>/apps.py
from django.apps import AppConfig

class IceCreamConfig(AppConfig):
    ...
    verbose_name = 'Каталог мороженого'
```

Название модели: в классе `Meta`, вложенном в модель, задайте необходимое название:

```
# <имя_приложения>/models.py
from django.db import models
...


class Category(models.Model):
    ...
    class Meta:
        verbose_name = 'категория'
        verbose_name_plural = 'Категории'
```

`verbose_name_plural` — название во множественном числе.

Название атрибута модели, отображаемое в админке и в формах, устанавливается через атрибут `verbose_name` в описании типа поля:

```
# <имя_приложения>/models.py
from django.db import models
...


class Category(models.Model):
```

```
    title = models.CharField(max_length=256, verbose_name='Название')
    slug = models.SlugField(max_length=64, unique=True, verbose_name='Сла
    output_order = models.PositiveSmallIntegerField(
        default=100,
        verbose_name='Порядок отображения'
    )
```

Читаемые названия объектов задаются в «магическом» методе `__str__`:

```
from django.db import models


class Category(models.Model):
    ...
    def __str__(self):
        return self.title
        # Теперь при попытке напечатать объект класса Category
        # на печать будет выведено значение его поля title.
```

По правилам оформления кода Django метод `__str__()` полагается определять после класса `Meta`.

Тонкая настройка админ-зоны

Настройки админ-зоны по умолчанию описаны в классе `ModelAdmin` приложения `admin`. Чтобы изменить интерфейс админки — следует переопределить дефолтные настройки этого класса:

1. Создать собственный класс, унаследовав его от `admin.ModelAdmin`.
2. Зарегистрировать этот класс как источник настроек админ-зоны.
3. В унаследованном классе переопределить настройки интерфейса админки.

```
# <имя_приложения>/admin.py
...
# Создаём класс, в котором будем описывать настройки админки:
class IceCreamAdmin(admin.ModelAdmin):
    # В этом классе опишем все настройки, какие захотим.
    ...
```

```
# Регистрируем новый класс:
# указываем, что для отображения админки модели IceCream
# вместо стандартного класса нужно нужно использовать класс IceCreamAdmin
admin.site.register(IceCream, IceCreamAdmin)
```

В классе `IceCreamAdmin` можно указать:

- какие поля будут показаны на странице списка объектов (атрибут `list_display`, это кортеж);
- какие поля можно редактировать прямо на странице списка объектов (атрибут `list_editable`, кортеж);
- `search_fields` — перечень полей, по которым будет проводиться поиск (кортеж). Форма поиска отображается над списком элементов;
- `list_filter` — кортеж с полями, по которым можно фильтровать записи. Фильтры отобразятся справа от списка элементов;
- `list_display_links` — кортеж с перечнем полей, при клике по которым можно перейти на страницу просмотра и редактирования записи. По умолчанию такой ссылкой служит первое отображаемое поле.

```
# <имя_приложения>/admin.py
...

class IceCreamAdmin(admin.ModelAdmin):
    list_display = (
        'title',
        'description',
        'is_published',
        'is_on_main',
        'category',
        'wrapper'
    )
    list_editable = (
        'is_published',
        'is_on_main',
        'category'
    )
    search_fields = ('title',)
```

```
    list_filter = ('category',)
    list_display_links = ('title',)

# Регистрируем кастомное представление админ-зоны для модели IceCream:
admin.site.register(IceCream, IceCreamAdmin)

...
```

Если в каких-то записях не задано значение необязательного поля — в админке на месте «пустого» значения будет отображён символ `-`. Вместо прочерка можно вывести любую строку: для этого надо переопределить атрибут `empty_value_display`.

```
# <имя_приложения>/admin.py
...

class IceCreamAdmin(admin.ModelAdmin):
    ...
    # Это свойство сработает для всех полей этой модели.
    # Вместо пустого значения будет выводиться строка "Не задано".
    empty_value_display = 'Не задано'
...
```

Интерфейс для управления записями, связанными «многие ко многим»

Интерфейс администратора можно настроить так, чтобы на странице редактирования определённой записи отображались связанные с ней записи другой модели.

Такие блоки называют «вставки», для их настройки в Django есть классы `admin.TabularInline` и `admin.StackedInline`.

`TabularInline` отображает поля вставки в строку, а `StackedInline` — столбом, одно под другим.

```
# <имя_приложения>/admin.py
...

# Подготавливаем модель IceCream для вставки на страницу другой модели.
```

```
class IceCreamInline(admin.StackedInline):
    model = IceCream
    extra = 0

class CategoryAdmin(admin.ModelAdmin):
    # Добавляем вставку на страницу управления объектом модели Category:
    inlines = (
        IceCreamInline,
    )
    list_display = (
        'title',
    )

# Регистрируем кастомное представление админ-зоны для модели Category:
admin.site.register(Category, CategoryAdmin)
```

Подсказки для полей модели

Подсказка указывается в поле модели, в именованном аргументе `help_text`:

```
# <имя_приложения>/models.py
from django.db import models


class Wrapper(models.Model):
    title = models.CharField(
        'Название',
        max_length=256,
        help_text='Уникальное название обёртки, не более 256 символов'
    )

    class Meta:
        verbose_name = 'обёртка'
        verbose_name_plural = 'Обёртки'

    def __str__(self):
        return self.title
```

Практикум