Тонкая настройка админ панели

Смотреть материал на видео

Архив проекта: lesson-24-coolsite.zip

На этом занятии мы сделаем некоторые изменения в админ-панели, настроим ее под свой разрабатываемый сайт. Если перейти в документацию по Django, то на официальной странице:

https://docs.djangoproject.com/en/3.1/

есть раздел «Admin» и ссылка «Admin site», где подробно объясняются различные нюансы настройки админ-панели. Это огромная тема, поэтому, мы коснемся лишь некоторых базовых вещей.

Меняем стили оформления

Вначале давайте посмотрим, как можно поменять стили оформления админ-панели. Это бывает необходимо, чтобы админка визуально выглядела в тех же тонах, что и основной сайт. Для этого,

мы с помощью панели Debug Toolbar посмотрим какие используются шаблоны для формирования этой страницы:

- admin/index.html

  D:\\Python\\django\\djsite\\venv\\lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\index.html

- admin/base_site.html

  D:\\Python\\django\\djsite\\venv\\lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\base_site.html

- admin/base.html

  D:\\Python\\django\\djsite\\venv\\lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\base.html

- admin/app_list.html

  D:\\Python\\django\\djsite\\venv\\lib\\site-packages\\django\\contrib\\admin\\templates\\admin\\app_list.html

Здесь нам показываются четыре шаблона и по названию можно понять, что базовым для страниц сайта является шаблон base_site.html. Также под ним прописан путь, где он располагается.

Давайте, откроем его и по идее, конечно, можно было бы внести изменения прямо здесь. Но это не лучшая практика. Все дополнительные файлы, которые были сформированы при установке Django, лучше оставлять без изменений. А переопределение делать уже непосредственно в нашем проекте. В частности, если добавить в каталог coolsite/templates подкаталог admin и там

разместить файл с тем же именем base_site.html, то Django, сканируя первым этот каталог, найдет первым файл admin/base_site.html и именно его будет использовать.

Создадим такой файл. Перейдем в браузер, обновим страницу админ-панели и не увидим никаких изменений. Мало того, если посмотреть в Debug Toolbar, то путь к шаблону base_site.html остался прежним. Что не так? Дело в том, что Django по умолчанию ищет шаблоны исключительно в папках приложений. У нас же каталог templates находится непосредственно в корне проекта, что необходимо, чтобы произошло переопределение шаблона base_site.html. Этот новый, нестандартный путь следует указать в файле конфигурации settings.py в коллекции TEMPLATES:

```
TEMPLATES = [
    {
        'BACKEND': 'django.templa
        'DIRS': [os.path.join(BAS
...
    },
]
```

Теперь, после перезапуска веб-сервера и обновлении админ-панели увидим пустую страницу, т.к. именно наш шаблон сейчас был взят. Скопируем в него содержимое из аналогичного файла админки и при обновлении видим все ту же админку.

В файле base_site.html видим три блока:

- block title – отвечает за отображение заголовка во вкладке;
- block branding – за отображение ссылки «Администрирование Django» в верхней панели;
- block nav-global – глобальный блок навигации.

Сейчас мы хотим прописать свои собственные стили оформления. Как это сделать? Для этого откроем шаблон более верхнего уровня base.html и видим специальный блок:

{% block extrastyle %}{% endblock %}

который служит для добавления дополнительных стилей оформления. Пропишем его в файле base_site.html и в нем укажем свой файл оформления:

```
{% block extrastyle %}
<link rel="stylesheet" href="{% s
{% endblock %}
```

Соответственно, выше загрузим тег static:

```
{% load static %}
```

и сформируем файл admin.css в нашем приложении в каталоге static/css:

```
#header {
    background: #5e3a00;
}
```

Почему мы прописали именно такой селектор для изменения цвета верхней панели? Очень просто. Если перейти в браузер и проинспектировать этот элемент, то увидим тег div с id="header". Значит, для назначения свойств, можно использовать этот идентификатор. По аналогии можно

переобозначать стили для всех других элементов админ-панели. Например, такой же стиль пропишем и для заголовков блоков:

```
#header, .module caption {
    background: #5e3a00;
}
```

Я думаю, общий принцип понятен. Этим способом можно переопределить все стили и настроить вид админки под дизайн, цветовую схему вашего сайта.

Если же вам нужно изменить сам заголовок «Администрирование Django», то это лучше делать через модуль admin (файл women/admin.py), в котором, следует определить два таких атрибута:

```
admin.site.site_title = 'Админ-па
admin.site.site_header = 'Админ-п
```

Полный список подобных атрибутов можно посмотреть на странице документации:

https://docs.djangoproject.com/en/3.1/ref/contrib/admin/

Добавляем отображение изображений для постов в списке

Следующим шагом добавим отображение миниатюр изображений, связанных с нашими постами, непосредственно в списке. Сейчас у нас происходит отображение пути к изображению в колонке «Фото». Мы же хотим показывать непосредственно изображение, а не путь. Как это сделать?

Для этого в классе WomenAdmin (файл women/admin.py) следует прописать метод, который бы возвращал HTML-код. И этот HTML-фрагмент, затем, подставим вместо путей. Имя метода мы придумываем сами, например, так:

```
    def get_html_photo(self, obje
        return mark_safe(f"<img s
```

Этот метод будет принимать второй параметр object – ссылку на текущую запись (на текущий пост) и, затем, возвращаем фрагмент HTML, но помеченный фильтром safe, чтобы теги воспринимались как теги и не экранировались, то есть, не заменялись бы спецсимволами. Именно для этого и используется функция mark_safe(). Все, если теперь вместо поля photo (в списке list_display) указать метод get_html_photo:

```
list_display = ('id', 'title', 't
```

должны увидеть вместо маршрутов картинки, связанные с постами. Однако, у нас отображается ошибка с исключением ValueError, так как не все посты имеют изображения. Поэтому в методе get_html_photo следует прописать проверку:

```
    def get_html_photo(self, obje
        if object.photo:
            return mark_safe(f"<i
```

Причем, по else можно ничего не возвращать (будет формироваться значение None), тогда Django автоматически поставит дефис там, где фотография отсутствует. Если же прописать иначе:

```
    def get_html_photo(self, obje
        if object.photo:
            return mark_safe(f"<i
        else:
            return "Нет фото"
```

То вместо дефиса увидим нашу строку «Нет фото». Наконец, чтобы поменять название столбца (вместо «GET HTML PHOTO»), нужно у объекта-метода get_html_photo определить специальный атрибут:

```
get_html_photo.short_description
```

Этот пример с выводом фотографии показывает, как можно заменять стандартный вывод полей списка на свой собственный, определяя дополнительные специальные методы и указывая их в списке list_display. То есть, следуя этому принципу, мы можем формировать самую разную информацию для отображения в списках.

Наконец, мы можем сделать вывод фотографии и непосредственно на странице редактирования поста. Для этого, опять же в классе WomenAdmin (файл women/admin.py) пропишем еще один атрибут:

```
fields = ('title', 'slug', 'cat',
```

Этот атрибут содержит порядок и список редактируемых полей, которые следует отображать в форме редактирования. Если же дополнительно нужно показывать не редактируемые поля, то дополнительно нужно прописать атрибут:

```
readonly_fields = ('time_create',
```

И только после этого, их можно добавить в коллекцию fields:

```
fields = ('title', 'slug', 'cat',
```

Отлично, это мы сделали, но как отобразить миниатюру? Для этого достаточно указать ссылку на наш метод get_html_photo() в этих атрибутах:

```
fields = ('title', 'slug', 'cat',
readonly_fields = ('time_create',
```

Все, теперь мы видим изображение загруженной фотографии в форме редактирования.

Чтобы узнать о других возможностях настройки формы редактирования, на странице документации можно посмотреть список атрибутов для класса ModelAdmin. В частности, установка вот такого атрибута:

```
save_on_top = True
```

добавляет верхнюю панель для управления записью, что бывает очень удобно.

Вот так, можно настраивать стили и отображение содержимого админ-панели.

Видео по теме

траничная (пагинация)

#19. Регистрация пользователей на сайте

#20. Д
польз

← Предыдущая