Добавляем слаги (slug) к URL-адресам

Смотреть материал на видео

Архив проекта: lesson-12-coolsite.zip

На этом занятии мы сделаем отображение отдельных статей по их слагу (slug). Если кто не знает, то slug – это уникальный фрагмент URL-адреса, ассоциированный с конкретной записью и, обычно, состоит из набора маленьких латинских букв, цифр, символов подчеркивания и дефиса. Например, статья «Арифметические операции» на сайте https://proproprogs.ru доступна по следующему адресу:

https://proproprogs.ru/python_base/arifmeticheskiye-operatsii

slug

Здесь slug – это последние символы, по которым и выбирается данная страница из БД. Использование слагов – рекомендуемая практика в веб-программировании. Такие страницы лучше ранжируются поисковыми системами и понятнее конечному пользователю.

Давайте вначале сделаем отображение статей по их идентификатору, а затем, заменим адрес на слаг. У нас уже есть функция-заглушка show_post() в файле women/views.py. Мы ее перепишем, следующим образом:

```
def show_post(request, post_id):
    post = get_object_or_404(Women, pk=post_id)

    context = {
        'post': post,
        'menu': menu,
        'title': post.title,
        'cat_selected': 1,
    }

    return render(request, 'women/post.html', context=context)
```

Здесь функция get_object_or_404 выбирает одну запись из таблицы Women, которая имеет идентификатор, равный post_id, либо генерирует исключение 404, если запись не была найдена. Это довольно частая операция, когда нужно найти какую-либо отдельную запись, а в противном случае, перенаправить пользователя на заготовленную страницу 404. Поэтому в Django для таких случаев заготовлена специальная функция.

Далее, формируется словарь из параметров шаблона и отображается страница на основе шаблона post.html. У нас пока нет такого файла, добавим его со следующим содержимым:

```
{% extends 'women/base.html' %}

{% block content %}

<h1>{{post.title}}</h1>

{% if post.photo %}
<p><img class="img-article-left" src="{{post.photo.url}}"></p>
{% endif %}

{{post.content|linebreaks}}
{% endblock %}
```

Здесь все достаточно очевидно. Вначале отображаем заголовок h1, затем, фотографию статьи, если она есть, ну и потом уже содержимое самой статьи.

Если теперь перейти по ссылке, то увидим полноценную статью. Если же указать неверный адрес, то получим исключение 404. Повторю еще раз, исключения в таком развернутом виде отображаются только в режиме отладки сайта. При эксплуатации с константой DEBUG = False вместо исключения отображается заготовленная страница 404.

Добавление слага

Следующим шагом сделаем отображение статей по их слагу. Но откуда нам его взять? Для этого в модели Women необходимо прописать еще одно поле, которое так и назовем – slug:

```
class Women(models.Model):
    title = models.CharField(max_length=255, verbose_name="Заголовок")
    slug = models.SlugField(max_length=255, unique=True, db_index=True, verbose_name="URL")
...
```

Я его определил после поля title, указал уникальным, индексируемым и имя URL, отображаемое в админке. Однако, если сейчас попытаться создать миграцию для внесения этих изменений в структуру таблицы women:

python manage.py makemigrations

то увидим предупреждение, что поле не может быть пустым (так как у нас есть записи в таблице). Чтобы таблицы были сформированы как надо, я решил создать БД заново. Поэтому сразу добавил такое же поле в модели Category:

```
class Category(models.Model):
    name = models.CharField(max_length=100, db_index=True, verbose_name="Категория")
    slug = models.SlugField(max_length=255, unique=True, db_index=True, verbose_name="URL")
...
```

Удалим все файлы миграций, прежний файл БД и выполним команду

python manage.py makemigrations

для создания первой мигации. Затем, с помощью команды:

python manage.py migrate

сформируем таблицы с новыми структурами. Этот пример хорошо показывает, как важно заранее продумывать структуры таблиц для сайта. Осталось восстановить записи в БД. Для этого я заново создам суперпользователя для админки:

python manage.py createsuperuser

с именем root, почтой root@coolsite.ru и паролем 1234. Запускаем веб-сервер и заходим в админ-панель.

Для начала добавим категории. Здесь нам предлагается ввести ее название и слаг (URL). Конечно, можно заполнить оба поля вручную, например, «Актрисы» и «actrisi». Но, так как слаг, обычно, повторяет заголовок, только записанный латиницей, то фреймворк Django позволяет этот процесс автоматизировать. Давайте откроем файл women/admin.py и для модели Category в классе CategoryAdmin добавим атрибут:

```
prepopulated_fields = {"slug": ("name", )}
```

Это специальное свойство, которое указывает фреймворку автоматически заполнять поле slug по данным поля name.

Возвращаемся в админку, обновляем страницу и, смотрите, при вводе строки в поле name, автоматически формируется поле slug. Это очень здорово и значительно облегчает нашу работу. Теперь можно совершенно спокойно добавить две рубрики «Актрисы» и «Певицы».

Далее, прежде чем добавлять статьи, сделаем такую же связку по слагу для модели Women в классе WomenAdmin:

```
prepopulated_fields = {"slug": ("title",)}
```

только здесь мы указываем поле title. Возвращаемся в админ-панель и на вкладке добавления женщин введем информацию по актрисам:

Анджелина Джоли, Дженнифер Лоуренс, Джулия Робертс, Марго Робби, Ума Турман

А также по певицам:

Ариана Гранде, Бейонсе, Кэтти Перри, Рианна, Шакира

Отлично, база данных готова и теперь можно сделать отображение статей по слагу. Для этого откроем файл women/urls.py и в списке urlpatterns изменим маршрут для постов на следующий:

```
path('post/<slug:post_slug>/', show_post, name='post'),
```

Затем, в файле women/views.py немного поменяем функцию представления show_post:

```
def show_post(request, post_slug):
    post = get_object_or_404(Women, slug=post_slug)
...
```

И в модели Women (в файле women/models.py) будем формировать URL-адрес по параметру slug:

```
class Women(models.Model):
...
    def get_absolute_url(self):
        return reverse('post', kwargs={'post_slug': self.slug})
...
```

Все, обновляем главную страницу сайта и видим, что теперь посты доступны по слагу, а не идентификатору. Этот пример показывает как в Django легко и просто можно менять URL-адреса и вместо id использовать другие поля, в частности, слаг. При этом, мы не производили совершенно никаких изменений в шаблонах, благодаря использованию метода get_absolute_url() в модели Women. Кроме того, Django автоматически защищает такие адреса от SQL-инъекций, когда злоумышленник пытается выполнить SQL-запрос, прописывая его в адресной строке браузера. Благодаря всем этим мелочам, которые берет на себя фреймворк, даже начинающий веб-мастер может конструировать вполне безопасные сайты с богатым функционалом.

Аналогичную операцию использования слагов можно сделать и для отображения рубрик. Предлагаю вам выполнить это самостоятельно для закрепления материала.

Видео по теме

#1. Django - что это такое, порядок установки

#2. Модель MTV. Маршрутизация. Функции представления

#3. Маршрутизация, обработка исключений запросов,

#4. Определение моделей. Миграции: создание и выполнение

#5. CRUD - основы ORM по работе с моделями

#6. Шаблоны (templates). Начало

#7. Подкл статически Фильтры ш

← Предыдущая Следующая →