Классы представлений, регистрация, оптимизация

Классы представлений: ListView, DetailView, CreateView

Основы ORM Django за час

Mixins - убираем дублирование кода

Постраничная навигация (пагинация)

Регистрация пользователей на сайте

Делаем авторизацию пользователей на сайте

Оптимизация сайта с Django Debug Toolbar

Включаем кэширование данных

Использование капчи captcha

Тонкая настройка админ панели

Поделиться

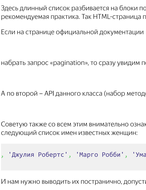
Наш канал

Постраничная навигация (пагинация)

Смотреть материал на видео

Архив проекта: lesson-18-cookiste.zip

На этом занятии продолжим совершенствовать наш учебный сайт и рассмотрим следующий вопрос – постраничную навигацию. Иногда еще говорят «пагинация» (от лат. pagina – страница). Но это все о том, как на сайте представлять длинные списки чего-либо. Вот классический пример постраничной навигации:



Здесь длинный список разбивается на блоки по 10 задач и внизу следует номера страниц для отображения остальных номеров. Это, в некотором смысле, рекомендуемая практика. Так HTML-страница получается не слишком большой по объему и пользователю удобнее ориентироваться в данных списка.

Если на странице официальной документации

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

набрать запрос «pagination», то сразу увидим первые две подходящие ссылки. По первой ссылке представлен пример работы класса пагинатора:

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

А по второй – API данного класса (набор методов и свойств для работы с ним):

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

Советую также со всем этим внимательно ознакомиться. А мы рассмотрим пример работы класса Paginator на конкретном примере. Предположим, имеется следующий список имен известных женщин:

```
['Джулия Робертс', 'Марго Робби', 'Ума Турман', 'Ариана Гранде', 'Бейонсе', 'Кэтти Перри', 'Рианна', 'Шакира']
```

И нам нужно выводить их постранично, допустим, по три имени на странице. Для этого, вначале, выполним импорт класса:

```
from django.core.paginator import Paginator
```

и создадим его экземпляр:

```
p = Paginator(women, 3)
```

Все, наш пагинатор готов к использованию. Например:

```
p.count # число элементов в списке
p.num_pages # число страниц (10:3 = 4 – округление до большего)
p.page_range # итератор для перебора номеров страниц
```

Чтобы получить первую страницу со списком имен, используется метод:

```
p1 = p.page(1) # получение первой страницы
p1.object_list # список элементов текущей страницы
p1.has_next() # имеется ли следующая страница
p1.has_previous() # имеется ли предыдущая страница
p1.has_other_pages() # имеется ли вообще страницы
p1.next_page_number() # номер следующей страницы
p1.previous_page_number() # номер предыдущей страницы
```

Я не буду приводить здесь полный список свойств и методов объекта пагинации, вы все это легко сможете посмотреть на странице документации по указанным ссылкам. Лучше давайте посмотрим, как все это можно использовать непосредственно на сайте.

Как вы помните, за отображение списка у нас отвечает базовый класс ListView. В этот класс уже встроен механизм постраничного вывода списков. Согласно документации, все что нам нужно, это определить атрибут:

paginate_by = N

указав число N – количество элементов на одной странице.

При использовании функций представлений, нужно уже самостоятельно создавать объект класса Paginator, получать номер текущей страницы и для нее формировать объект с набором элементов. Ниже в документации это все приведено, поэтому, здесь я покажу короткий пример, как можно этим воспользоваться в рамках нашего сайта.

Смотрите, у нас есть функция представления about(). Давайте в нее добавим следующие строчки:

```
    contact_list = Women.objects.all()
    paginator = Paginator(contact_list, 3)

    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
```

Я их, фактически, скопировал из документации, прописал нашу модель Women и указал отображать 3 элемента на странице. Далее, коллекцию page_obj для текущей страницы нужно передать в шаблон:

```
return render(request, 'women/about.html', {'page_obj': page_obj, 'menu': menu, 'title': 'О сайте'})
```

А в самом шаблоне about.html, перебрать эту коллекцию и отобразить на странице:

```
{% for contact in page_obj %}
<p>{{ contact }}</p>
{% endfor %}
```

Как видите, все предельно просто. Если теперь открыть страницу «О сайте», то увидим первые три имени из модели Women. Чтобы перейти на следующую страницу, нужно в запросе указать специальный параметр page с номером страницы, например, так:

http://127.0.0.1:8000/about/?page=2

Именно этот параметр читается из словаря GET и подставляется в метод get_page для получения соответствующей страницы. Причем, указывая не существующий номер, будет отображаться последняя страница. Если вместо номера указать, например, буквы, то параметр игнорируется и отображается первая страница.

Осталось обработать ссылки на номера страниц непосредственно в шаблоне. Для этого мы через объект page_obj обратимся к объекту paginator и у него укажем метод page_range():

page_obj.paginator.page_range

Обратите внимание, в шаблонах у методов мы не пишем круглые скобки, весь вызов берет на себя шаблонизатор. Итак, метод page_range() возвращает итератор для перебора номеров страниц. Поэтому в шаблоне мы можем воспользоваться тегом for для формирования номеров страниц:

```
<nav>
    <ul>
        {% for p in page_obj.paginator.page_range %}
        <li>
            <a href="?page={{ p }}">{{ p }}</a>
        </li>
        {% endfor %}
    </ul>
</nav>
```

Здесь все достаточно очевидно, мы перебираем доступные номера, формируем ссылку с параметром page и текущим номером. Именем ссылки также выступает ее номер. Если теперь перейти в браузер и обновить страницу, то увидим номера страниц.

Я здесь ничего улучшать пока не буду, это был пример того, как можно использовать пагинацию совместно с функциями представлений. Давайте теперь перейдем к классам и посмотрим как подобный функционал реализовать на главной странице.

Для этого в дочерним классе WomenHome достаточно прописать атрибут:

```
paginate_by = 3
```

Получим разбивку по три поста на страницу. Если теперь открыть главную страницу, то именно первые три записи мы и увидим. Но у нас пока нет отображения номеров страниц. Давайте добавим их. Для этого на уровне шаблонов у нас появляется специальные объекты paginator и page_obj. Воспользуемся объектом paginator и отобразим в шаблоне список страниц. Я это сделаю в файле base.html, чтобы страницы отображались во всех шаблонах, если они есть. Значит, в файле base.html после блока контента {% block content %}{% endblock %}} прогишем следующие уже знакомые нам строчки:

```
<nav class="list-pages">
    <ul>
        {% for p in paginator.page_range %}
        <li class="page-num">
            <a href="?page={{ p }}">{{ p }}</a>
        </li>
        {% endfor %}
    </ul>
</nav>
```

Только здесь, в отличие от шаблона about.html, мы используем объект paginator напрямую, т.к. класс ListView его автоматически добавляет в шаблон. Из файла about.html уберем вывод списка страниц, т.к. теперь в этом нет необходимости.

Отлично, если сейчас обновить главную страницу сайта, то увидим номера страниц в виде списка вместе со стилями, которые подключены через классы оформлений list-pages и page-num. Сами стили я дополнительно определил в файле styles.css и, если интересно, то вы сможете их самостоятельно посмотреть в проекте, скачав его с github.

Итак, главная страница отображается в нужном нам виде, но если открыть отдельные рубрики, то пагинация пропадает. Почему? Если мы вернемся в проект, то увидим, что за это отвечает отдельный класс WomenCategory, в котором, по идее, также нужно прописать

paginate_by = 3

Однако, это будет дублирование кода и нам проще эту строчку записать в общем классе DataMixin:

```
class DataMixin:
    paginate_by = 3
...
```

А из класса WomenHome ее убрать. Теперь, на всех страницах, унаследованных от ListView, мы будем видеть автоматическую пагинацию. Правда, везде будет отображаться по три статьи. В данном случае нас это устраивает, но если на разных страницах нужно разное число рубрик при пагинации, то, конечно, в каждом классе отдельно следует прописать атрибут paginate_by со своим значением.

Однако, если сейчас в paginate_by указать большое число, например, 30, то на главной странице будет отображаться внизу один единственный номер, что не очень хорошо. Было бы лучше, при одной единственной странице вообще не выводить никаких номеров. Для этого, в шаблоне base.html вывод пагинации будем осуществлять по следующему условию:

```
{% if page_obj.has_other_pages %}
<nav class="list-pages">
...
</nav>
{% endif %}
```

Все, теперь в базовом варианте отображение номеров страниц сделано. Далее, сделаем некоторые улучшения. Например, нам следовало бы убирать ссылку на текущий номер страницы. Для этого в шаблоне base.html мы, при выводе ссылок, будем проверять, если это текущий номер, то отображать его как текст:

```
<nav class="list-pages">
    <ul>
        {% for p in paginator.page_range %}
            {% if page_obj.number == p %}
        <li class="page-num page-num-selected">{{ p }}</li>
            {% else %}
        <li class="page-num">
            <a href="?page={{ p }}">{{ p }}</a>
        </li>
            {% endif %}
        {% endfor %}
    </ul>
</nav>
```

Здесь используется свойство number объекта page_obj для определения номера текущей страницы, и если этот номер равен p, то его следует отобразить как текст. Иначе, идет отображение в виде ссылки.

Следующая проблема, с которой мы можем столкнуться – это большое число номеров при большом числе записей. И при навигации это будет приводить к некоторым неудобствам, да и на дизайне страницы большое число номеров оказывается не лучшим образом. Поэтому, на практике, ограничиваются небольшим числом отображаемых номеров слева и справа от текущего номера. Вот это мы сейчас и сделаем.

Для начала, в админ-панели отметим все статьи для публикации (чтобы номеров страниц было побольше) и в классе DataMixin установим атрибут:

```
paginate_by = 2
```

Теперь на главной странице у нас появляется семь номеров. Но, мы решили отображать не все их, а, допустим, два справа и два слева:

Этот диапазон, на уровне шаблонов, можно определить через фильтр add, уменьшая значение на 2 и увеличивая на 2 относительно текущего номера страницы. Поэтому в цикле, при отображении этих номеров, мы можем записать следующее условие вывода:

```
        {% for p in paginator.page_range %}
            {% if page_obj.number == p %}
        <li class="page-num page-num-selected">{{ p }}</li>
            {% elif p >= page_obj.number|add:-2 and p <= page_obj.number|add:2  %}
        <li class="page-num">
            <a href="?page={{ p }}">{{ p }}</a>
        </li>
            {% endif %}
        {% endfor %}
```

Все, теперь, обновляя страницу сайта, мы увидим по два номера слева и справа.

Следующим нашим улучшением будет отображение ссылок для перехода на предыдущую и следующую страницы. Это делается очень просто. В шаблоне base.html до цикла запишем такие строчки:

```
{% if page_obj.has_previous %}
<li class="page-num">
        <a href="?page={{ page_obj.previous_page_number }}">&lt;</a>
</li>
{% endif %}
```

Мы здесь вначале проверяем, имеет ли смысл вообще выводить ссылку на предыдущую страницу (для первых номеров это не нужно делать). И, если она нужна, то отображаем как элемент списка с номером страницы согласно атрибуту previous_page_number.

То же самое делаем и для отображения ссылки на следующую страницу. Только тепрь добавляем пункт после цикла for:

```
{% if page_obj.has_next %}
<li class="page-num">
        <a href="?page={{ page_obj.next_page_number }}">&gt;</a>
</li>
{% endif %}
```

Здесь также вначале идет проверка: нужно ли отображать ссылку. Если проверка проходит, то отображаем пункт списка со ссылкой согласно атрибуту next_page_number.

Все, можно обновить главную страницу сайта и увидим эти дополнительные элементы управления. Вот так, в базовом варианте, можно работать с пагинацией во фреймворке Django.

Видео по теме

← Предыдущая Следующая →