**Тема занятия № 16: Пагинация**

1. Класс Paginator: сам пагинатор. Создание пагинатора

При выводе большие списки практически всегда разбивают на отдельные пронумерованные части, включающие не более определенного количества позиций. Это позволяет уменьшить размер страниц и ускорить их загрузку. Для перехода на нужную часть списка на страницах создают набор гиперссылок.

Разбиением списков на части занимается программный механизм, носящий название пагинатора.

На заметку!

Пагинатор явно применяется только в контроллерах-функциях и в контроллерах-классах самого низкого уровня. Высокоуровневые контроллеры-классы, наподобие Listview, используют пагинатор неявно.

Класс Paginator из модуля django.core.paginator представляет сам пагинатор. Экземпляр именно этого класса необходимо создать, чтобы реализовать пагинацию.

Формат его конструктора:

Paginator(<набор записей>, <количество записей в части>[, orphans=0][,Allow\_empty\_first\_page=True])

Первый параметр задает набор записей, который должен разбиваться на части, второй — количество записей в части.

Необязательный параметр orphans указывает минимальное количество записей, которые могут присутствовать в последней части пагинатора. Если последняя часть пагинатора содержит меньше записей, то все эти записи будут выведены в составе предыдущей части. Если задать значение 0, то в последней части может присутствовать сколько угодно записей (поведение по умолчанию).

Необязательный параметр aiiow\_empty\_first\_page указывает, будет ли создаваться “Пустая” часть, если набор не содержит записей. Значение True разрешает это делать (поведение по умолчанию), значение False, напротив, предписывает возбудить в таком случае исключение emptypage ИЗ модуля django.core.paginator.

Класс Paginator поддерживает три атрибута:

□ count — общее количество записей во всех частях пагинатора;

□ num pages — количество частей, на которые разбит набор записей;

□ page range— итератор, последовательно возвращающий номера всех частей пагинатора, начиная с 1.

Однако для нас полезнее будут два метода этого класса:

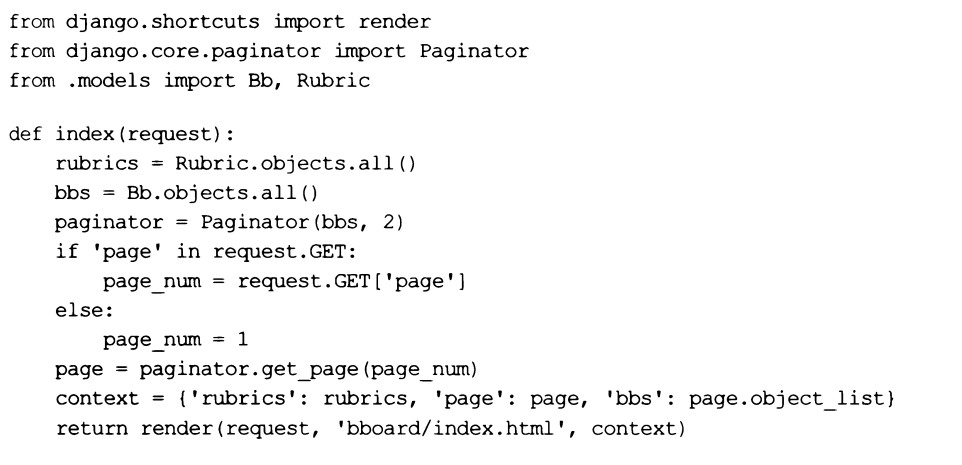
□ get\_page (<номер части>) — возвращает экземпляр класса Раде (он будет описан далее), представляющий часть с указанным номером. Нумерация частей начинается с 1.

Если номер части не является целочисленной величиной, то возвращается первая страница. Если номер части является отрицательным числом или превышает общее количество частей в пагинаторе, то возвращается последняя часть. Если получаемая часть "пуста", а при создании экземпляра класса Paginator параметру Aiiow enpty first page было дано значение False, возбуждается исключение Emptypage;

□ page(<номер части>) — то же самое, что и get\_page(), но в любом случае, если номер части не целое число, либо отрицательное число, либо оно превышает общее количество частей в пагинаторе, то возбуждается исключение Pagenotanlnteger ИЗ модуля dj ango. Core. Paginator.

Этот метод оставлен для совместимости с предыдущими версиями Django.

Пример использования пагинатора. В нем приведен код контроллера-функции index о, которая выводит список объявлений с разбиением на части (подразумевается, что номер части передается через GET-параметр раgе).



Мы проверяем, присутствует ли в наборе GET-параметров, полученных в запросе, параметр раде. Если это так, извлекаем из него номер части, которую нужно вывести на странице. В противном случае подразумевается, что посетитель запрашивает первую часть, которую мы и выводим.

В контексте шаблона создаем переменную bbs, которой присваиваем список записей, входящих в запрошенную часть (его можно извлечь из атрибута object list части пагинатора). Это позволит использовать уже имеющийся шаблон bboard\Index.html.

1. Класс Page: часть пагинатора. Вывод пагинатора

Класс Раде из модуля django.core.paginator представляет отдельную часть пагинатора, возвращенную методом get раде о ИЛИ раде ().

Класс Раде поддерживает следующие атрибуты:

□ object list —список записей, входящих в состав текущей части;

□ number — порядковый номер текущей части пагинатора (нумерация частей начинается с 1);

□ paginator— пагинатор (в виде экземпляра класса Paginator), создавший эту часть.

А вот набор методов этого класса:

□ has next () — возвращает True, если существуют следующие части пагинатора, и False — в противном случае;

□ has previous о — возвращает True, если существуют предыдущие части пагинатора, и False — в противном случае;

□ has other pages () — возвращает True, если существуют предыдущие или следующие части пагинатора, и False — в противном случае;

□ next page number о — возвращает номер следующей части пагинатора. Если это последняя часть (т. Е. Следующей части не существует), то возбуждает исключение emptypage;

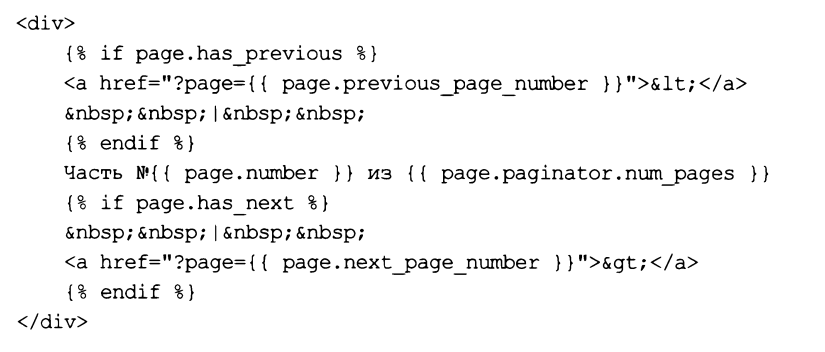
□ previous page number о — возвращает номер предыдущей части пагинатора.

Если это первая часть (т. Е. Предыдущей части не существует), то возбуждает исключение emptypage;

□ start index () — возвращает порядковый номер первой записи, присутствующей в текущей части. Нумерация записей в этом случае начинается с 1;

□ end indexo — возвращает порядковый номер последней записи, присутствующей в текущей части. Нумерация записей в этом случае начинается с 1.

Далее приведен фрагмент кода шаблона bboard\index.html, выводящий набор гиперссылок для перехода между частями пагинатора:

****