**Тема занятия № 15: Шаблонизатор. Обработка статических файлов**

1. Теги шаблонизатора

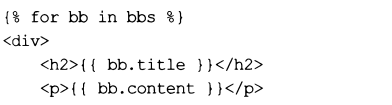
Теги шаблонизатора управляют генерированием содержимого страницы. Они заключаются в последовательности символов {% и %}. Как и HTML-теги, теги шаблонизатора бывают одинарными и парными.

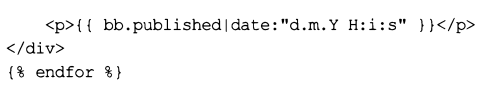
Одинарный тег, как правило, выводит на страницу какое-либо значение, вычисляемое самим фреймворком. Пример одинарного тега csrf token, выводящего электронный жетон, который используется подсистемой безопасности фреймворка:

****

Парный тег “охватывает” фрагмент кода и выполняет над ним какие-либо действия. Он фактически состоит из двух тегов: открывающего, помечающего начало “охватываемого” фрагмента (содержимого), и закрывающего, который помечает его конец. Закрывающий тег имеет то же имя, что и открывающий, но с добавленным впереди префиксом end.

Например, парный тег for ... Endfor повторяет содержащийся в нем фрагмент столько раз, сколько элементов находится в указанной в этом теге последовательности. Тег for —открывающий, а тег endfor —закрывающий:



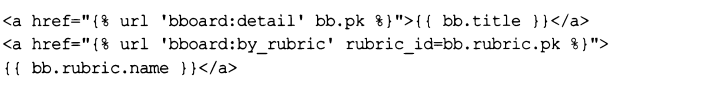


Теги, поддерживаемые шаблонизатором Django:

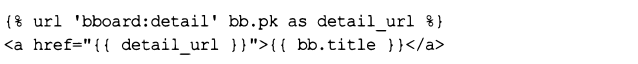
□ url —формирует интернет-адрес путем обратного разрешения:



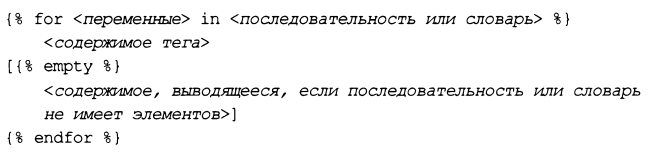
В имени маршрута при необходимости можно указать пространство имен. Параметры в списке могут быть как позиционными, так и именованными. Примеры:

****

По умолчанию тег вставляет сформированный адрес в шаблон. Но можно указать шаблонизатору сохранить адрес в переменной, записав ее после ключевого слова as. Пример:

****

□ for . . . Endfor — перебирает в цикле элементы указанной последовательности (или словаря) и для каждого элемента создает в коде страницы копию своего содержимого. Аналог цикла for ... In языка Python. Формат записи тега:



Значение очередного элемента последовательности заносится в указанную переменную, если же перебирается словарь, ТО МОЖНО указать две переменные под ключ и значение элемента соответственно. Эти переменные можно использовать в содержимом тега.

Также в содержимом присутствует следующий набор переменных, создаваемых самим тегом:

• forloop, counter— номер текущей итерации цикла (нумерация начинается Cl);

• forioop.countero— номер текущей итерации цикла (нумерация начинается С 0);

• forioop. Revcounter — число оставшихся итераций цикла (нумерация начинается с 1);

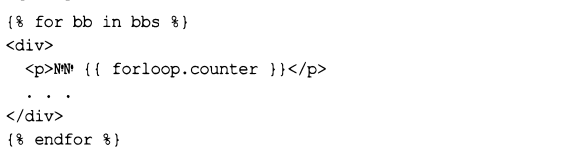
• forioop. Revcountero— число оставшихся итераций цикла (нумерация начинается с 0);

• forioop. First —True, если это первая итерация цикла, False, если не первая;

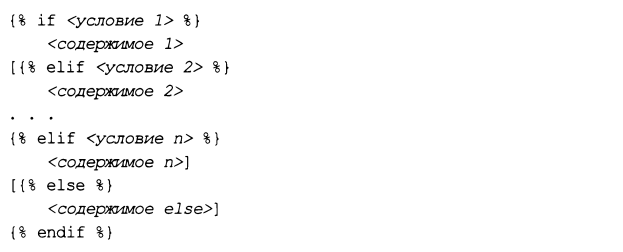
• forioop.last— True, если это последняя итерация цикла, False, если не последняя;

• forioop. Parentloop— применяется во вложенном цикле и хранит ссылку на “Внешний” цикл. Пример использования: forioop.parentloop.counter (получение номера текущей итерации “внешнего” цикла).

Пример:

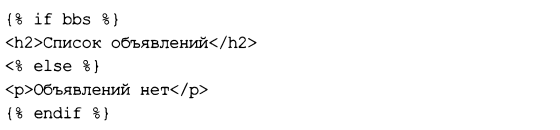
****

□ if . . . Elif . . . Else . . . Endif —аналог условного выражения Python:

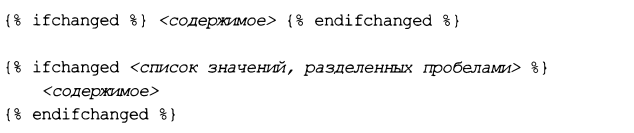


В условиях можно указывать операторы сравнения ==, ! =, <, >, <=, >=, in, not in, Is и is not, логические операторы and, or и not.

Пример:

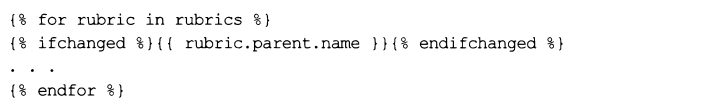


□ ifchanged . . . Endifchanged— применяется в циклах. Форматы использования:

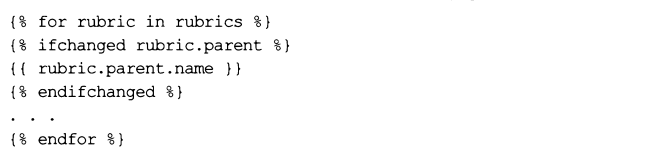
****

Первый формат выводит содержимое, если оно изменилось после предыдущей итерации цикла. Второй формат выводит содержимое, если изменилось одно из значений, приведенных в списке.

Выводим название рубрики, в которую вложена текущая рубрика, только если это название изменилось (т. Е. Если текущая рубрика вложена в другую рубрику, нежели предыдущая):

****

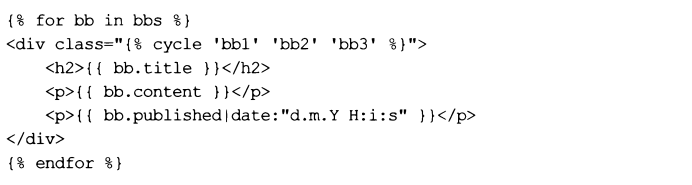
To же самое, только с использованием второго формата записи тега:

****

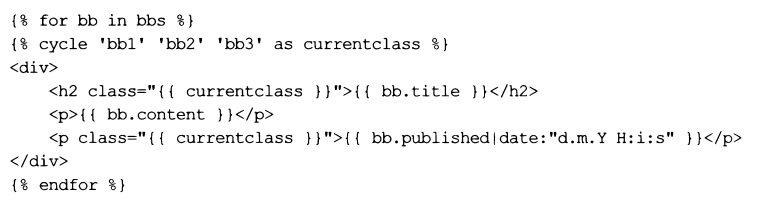
□ cycle — последовательно помещает в шаблон очередное значение из указанного перечня, Cycle <перечень значений, разделенных пробелами> [as <переменная>]

По достижении конца перечня перебор начинается с начала. Количество значений в перечне не ограничено.

В следующем примере при каждом проходе цикла for . . . In к блоку будут последовательно привязываться стилевые классы ы, Ь2, ьз, потом снова ы, Ь2 и т. д.:



Текущее значение перечня может быть занесено в переменную. Указанную после ключевого слова as, и вставлено в другом месте страницы:

****

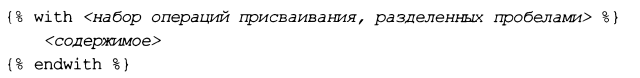
□ (% resetcycle [<переменная>] %} — сбрасывает тег cycle, после чего тот начинает перебирать указанные в нем значения с начала. По умолчанию сбрасывает последний тег cycle, записанный в шаблоне. Если же нужно сбросить конкретный тег, то следует указать переменную, записанную в нужном теге cycle после ключевого слова as;

О {% firstof Сперечень значений, разделенных пробелами> %} — помещает в шаблон первое из приведенных в перечне значений, не равное False (т. Е. Не ’’пустое”).

Следующий пример помещает на страницу либо значение поля phone объявления ьь; либо, если оно “пусто”, значение поля email; либо, если и оно не заполнено, Строку “На деревню дедушке";



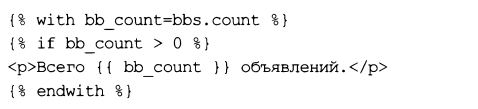
□ with . . . Endwith— заносит какие-либо значения в переменные и делает их доступными внутри своего содержимого:



Операции присваивания записываются так же, как и в Python.

Может использоваться для временного сохранения в переменных результатов каких-либо вычислений (например, полученных при обращении к методу класса) — чтобы потом не выполнять эти вычисления повторно.

Пример:

****

□ regroup— выполняет группировку указанной последовательности словарей ИЛИ объектов по значению элемента с заданным ключом или атрибута с заданным именем И помещает результат В переменную:

{% regroup <последовательность> by

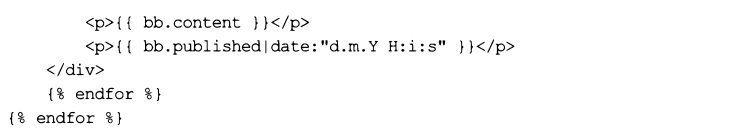
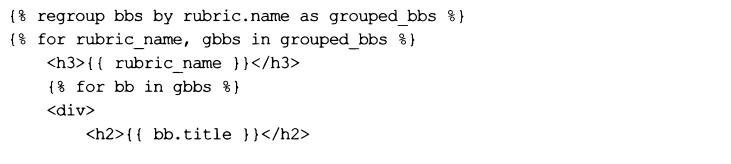
<ключ элемента или атрибут объекта> as <переменная> %}

Сохраненный в переменной результат представляет собой список объектов типа Namedtuple с элементами:

• grouper — значение элемента или атрибута, по которому выполнялась группировка;

• list — список словарей или объектов, относящихся к созданной группе.

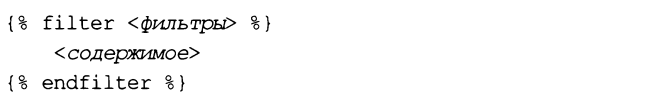
Пример группировки объявлений по рубрике:



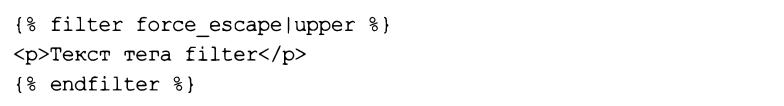
□ {% now <формат> %} — выводит текущие дату и время, оформленные согласно заданному формату (форматирование значений даты и времени описано далее — при рассмотрении фильтра date):



□ filter . . . Endfilter —применяет к содержимому указанные фильтры:

****

Применяем к абзацу фильтры force escape и upper;

****

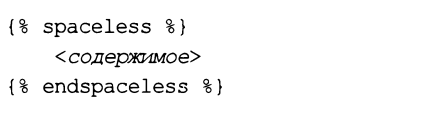
□ csrf token— выводит электронный жетон, используемый подсистемой безопасности Django. Применяется исключительно в веб-формах;

□ autoescape on | off . . . Endautoescape — включает ИЛИ отключает В содержимом автоматическое преобразование недопустимых знаков HTML (двойной кавычки, знаков ’’меньше” и ’’больше”) в соответствующие специальные символы при их выводе:

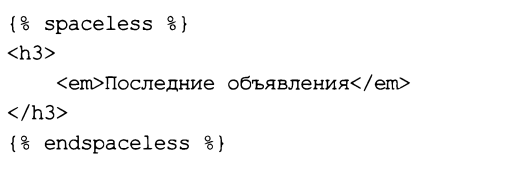
****

Значение on включает автоматическое преобразование, значение off — отключает;

□ spaceless . . . Endspaceless — удаляет в содержимом пробельные символы (в число которых входят пробел, табуляция, возврат каретки и перевод строки) между тегами:

****

Пример:

****

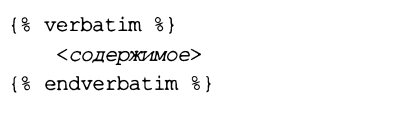
В результате в код страницы будет помещен фрагмент:

<ЬЗ><ет>Последние объявления</ет></113> О {% templatetag обозначение последовательности символов> %} — ВЫВОДИТ последовательность символов, которую иначе вывести не получится (примеры:

{{, }}, {%, %}). Поддерживаются следующие обозначения последовательностей символов:



□ verbatim . . . Endverbatin — ВЫВОДИТ содержимое как есть, не обрабатывая записанные в нем директивы, теги и фильтры шаблонизатора:

****

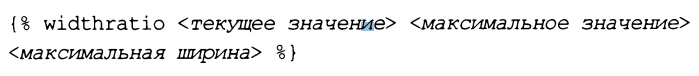
Пример:

****

\* {% load <псевдонимы библиотек тегов через пробел> %} — загружает библиотеки тегов с указанными псевдонимами:

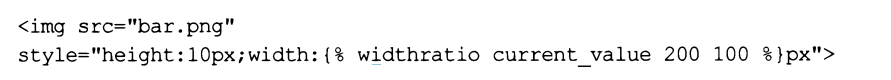
****

□ widthratio —применяется для создания диаграмм:

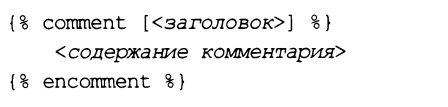
****

Текущее значение делится на максимальное значение, после чего получившееся частное умножается на максимальную ширину, и результат всего этого вставляется в шаблон.

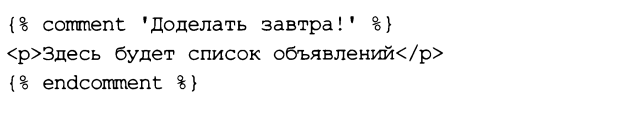
Пример использования этого довольно странного тега:



□ comment . . . Encomment — создает в коде шаблона комментарий, не обрабатываемый шаблонизатором:

****

У комментария можно задать необязательный заголовок. Пример:

****

Если комментарий занимает всего одну строку или часть ее, то его можно создать, заключив в последовательности символов {# и #}:

{# Не забыть сделать вывод рубрик! #}

□ {% debug %} — выводит разнообразную отладочную информацию, включая содержимое контекста шаблона и список отладочных модулей. Эта информация весьма объемна и не очень удобна на практике.

1. Фильтры

Фильтры шаблонизатора выполняют заданные преобразования значения перед его выводом.

Фильтр записывается непосредственно в директиве, после источника значения, и отделяется от него символом вертикальной черты (|). Пример:



Фильтры можно объединять, перечислив через вертикальную черту, в результате чего они будут обрабатываться последовательно, слева направо:



Вот список фильтров, поддерживаемых шаблонизатором Django:

□ date:<формат>— форматирует значение даты и времени согласно заданному формату. В строке формата допустимы следующие специальные символы:

• j — число без начального нуля;

• d —число с начальным нулем;

• m —номер месяца с начальным нулем;

• п —номер месяца без начального нуля;

• f и N — полное название месяца в именительном падеже с большой буквы;

• е — полное название месяца в родительном падеже и в нижнем регистре;

• м —сокращенное название месяца с большой буквы;

• ь —сокращенное название месяца в нижнем регистре;

• y и о — год из четырех цифр;

• у —год из двух цифр;

• ь — True, если это високосный год, False, если обычный;

• w —номер дня недели от 0 (воскресенье) до 6 (суббота);

• 1 — сокращенное название дня недели с большой буквы;

• d — полное название дня недели в именительном падеже с большой буквы;

• g — часы в 24-часовом формате без начального нуля;

• н — часы в 24-часовом формате с начальным нулем;

• g — часы в 12-часовом формате без начального нуля;

• h — часы в 12-часовом формате с начальным нулем;

• i — минуты;

• s — секунды с начальным нулем;

• и —микросекунды;

• а — обозначение половины суток в нижнем регистре (”д.п." или "п. П.");

• а— обозначение половины суток в верхнем регистре ("дп" или "пп");

• I — 1, если сейчас летнее время, 0, если зимнее;

• р— часы в 12-часовом формате и минуты. Если минуты равны 0, то они не указываются. Вместо 00:00 выводится строка "полночь", а вместо 12:00 — "полдень";

• f— часы в 12-часовом формате и минуты. Если минуты равны 0, то они не указываются;

• t —число дней в текущем месяце;

• z — порядковый номер дня в году;

• w — порядковый номер недели (неделя начинается с понедельника);

• е — название временной зоны;

• о — разница между текущим и гринвичским временем в часах;

• z — разница между текущим и гринвичским временем в секундах;

• с — дата и время в формате ISO 8601;

• г — дата и время в формате RFC 5322;

• и — время в формате UNIX (выражается как количество секунд, прошедших с полуночи 1 января 1970 года);

• т —название временной зоны, установленной в настройках компьютера.

Пример:



Также можно использовать следующие встроенные в Django форматы:

• date format — развернутый формат даты;

• datetime format — развернутый формат даты и времени;

• short date format — сокращенный формат даты;

• short datetime format — сокращенный формат даты и времени.

Пример:



□ timet-.<формат времени>] — форматирует выводимое значение времени согласно заданному формату. При написании формата применяются те же специальные символы, что и в случае фильтра date. Пример:



Для вывода времени с применением формата по умолчанию следует использовать обозначение time format:



Или вообще не указывать формат:



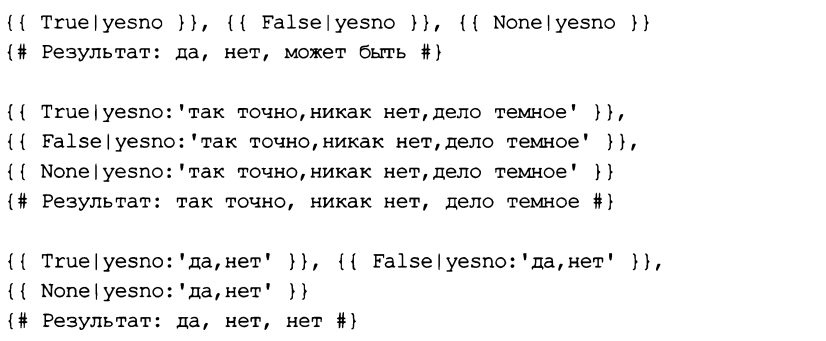
□ timesince [: <значение для сравнения>] — выводит промежуток времени, разделяющий выводимое значение даты и времени и заданное значение для сравнения, относящееся к будущему (если таковое не указано, в его качестве принимается сегодняшние дата и время). Результат выводится в виде, например, "3 недели, 6 дней", "6 дней 23 часа" и т. п. Если значение для сравнения ОТНОСИТСЯ к прошлому, выведет строку: "0 минут";

□ timeuntil [: <значение для сравнения>] — ТО же самое, ЧТО И timesince, НО значение для сравнения должно ОТНОСИТЬСЯ К Прошлому;

□ yesno [: <строка образцов>] — преобразует значения True, False и, возможно, None В слова "да", "нет" И "может быть".

Можно указать свои слова для преобразования, записав их в строке образцов вида <строка для True>,<строка для Faise>[,<cmpoka для None>]. Если строка для None не указана, то вместо нее будет выводиться строка для False (поскольку None будет неявно преобразовываться в False).

Примеры:

****

□ default: <величина>—если выводимое значение равно False, то возвращает указанную величину.

Следующий пример выведет строку "У товара нет цены", если поле price товара ьь не заполнено или хранит 0:



□ default\_if\_none: <величина>— ТО же самое, ЧТО И default, НО возвращает величину только в том случае, если выводимое значение равно None;

□ upper — переводит все буквы выводимого значения в верхний регистр;

□ lower —переводит все буквы выводимого значения в нижний регистр;

□ capfirst —переводит первую букву выводимого значения в верхний регистр;

□ title — переводит первую букву каждого слова в выводимом значении в верхний регистр;

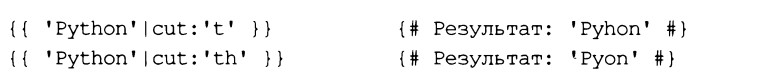
□ truncatechars: <длина>— обрезает выводимое значение до указанной длины, помещая в конец символ многоточия (...);

О truncatechars—html: <длина>— ТО же самое, ЧТО И truncatechars, НО сохраняет все HTML-теги, которые встретятся в выводимом значении;

□ truncatewords: количество слов>— обрезает выводимое значение, оставляя в нем указанное количество слов. В конце обрезанного значения помещается символ многоточия (...);

□ tгипсаtewords\_html: количество слов>— ТО же самое, ЧТО И truncatewords, НО сохраняет все HTML-теги, которые встретятся в выводимом значении;

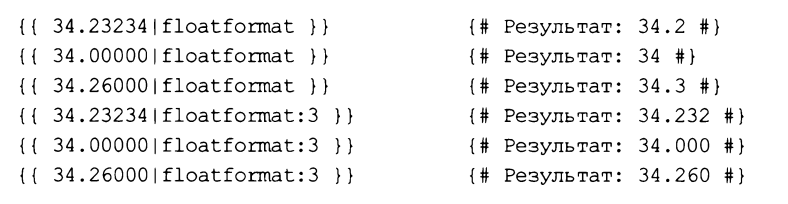
П wordwrap: <величина>— выполняет перенос выводимого строкового значения по словам таким образом, чтобы длина каждой получившейся в результате строки не превышала указанную величину, П cut: <удаляемая подстрока>— удаляет из выводимого значения все вхождения заданной подстроки:

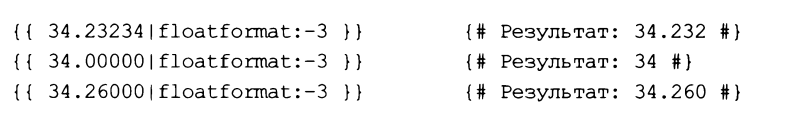


□ siugifу —преобразует выводимое строковое значение в слаг;

О stringformat :<формат>— форматирует выводимое значение согласно указанному формату. При написании формата применяются специальные символы, поддерживаемые оператором % языка Python (см. Страницу https://docs.python.org/3/library/stdtypes.html#old-string-formatting);

□ floatformat [: количество знаков после запятой>] — округляет выводимое вещественное число до заданного количества знаков после запятой. Целые числа автоматически приводятся к вещественному типу. Если указать положительное значение количества знаков, у преобразованных целых чисел дробная часть будет выводиться, если отрицательное — не будет. Значение количество знаков по умолчанию: -1. Примеры:





□ filesizeformat— выводит числовую величину как размер файла (примеры:

"100 байт", "8,8 КБ", "47,7 МБ");

□ add: <величина>-- Прибавляет К ВЫВОДИМОМУ Значению указанную величину.

Можно складывать числа, строки и последовательности;

□ divisibieby: <делитель>— возвращает True, если выводимое значение делится на указанный делитель без остатка, и False — в противном случае;

□ wordcount — возвращает число слов в выводимом строковом значении;

□ length— возвращает число элементов в выводимой последовательности. Также работает со строками;

□ iength\_is: <величина>— возвращает True, если длина выводимой последовательности равна указанной величине, и False — в противном случае;

□ first — возвращает первый элемент выводимой последовательности;

□ last — возвращает последний элемент выводимой последовательности;

□ random — возвращает случайный элемент выводимой последовательности;

□ slice:<оператор взятия среза Python>—возвращает срез выводимой последовательности. Оператор взятия среза записывается без квадратных скобок. Пример:

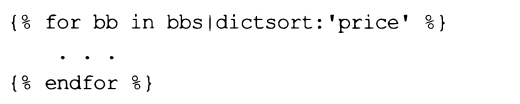
{{ rubric\_names|slice:'1:3’ }}

□ join: <разделитель>— возвращает строку, составленную из элементов выводимой последовательности, которые отделяются друг от друга разделителем;

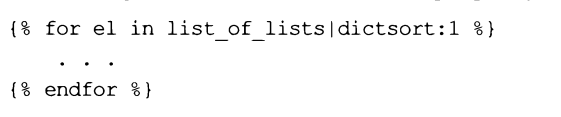
□ make list — преобразует выводимую строку в список, содержащий символы этой строки;

□ dictsort: <ключ элемента>— если выводимое значение представляет собой последовательность словарей или объектов, то сортирует ее по значениям элементов с указанным ключом. Сортировка выполняется по возрастанию значений.

Пример вывода объявлений с сортировкой по цене:



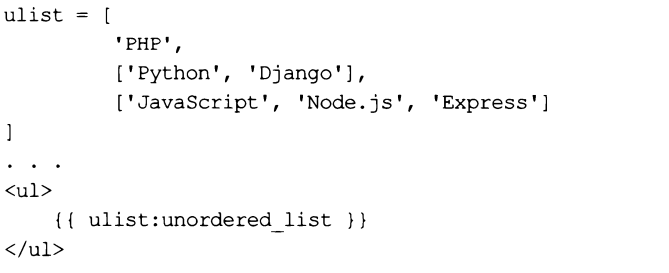
Можно сортировать последовательность списков или кортежей, только вместо ключа нужно указать индекс элемента вложенного списка (кортежа), по значениям которого следует выполнить сортировку. Пример:

****

□ dictsortreversed: <ключ элемента>— ТО же самое, ЧТО dictsort, ТОЛЬКО сортировка выполняется по убыванию значений;

□ unordered iist— используется, если выводимым значением является список или кортеж, элементы которого представлены также списками или кортежами.

Возвращает HTML-код, создающий набор вложенных друг в друга неупорядоченных списков, без “внешних” тегов <ui> и </ui>. Пример:

****

Выводимый результат:

****

□ linebreaksbr — заменяет в выводимом строковом значении все символы перевода строки на HTML-теги <Ьг>;

□ linebreaks— разбивает выводимое строковое значение на отдельные строки.

Если в значении встретится одинарный символ перевода строки, то он будет заменен HTML-тегом <Ьг>. Если встретится двойной символ перевода строки, то разделяемые им части значения будут заключены в теги <р>;

□ uriize— преобразует все встретившиеся в выводимом значении интернетадреса и адреса электронной почты в гиперссылки (теги <а>). В каждый тег, создающий обычную гиперссылку, добавляется атрибут rei со значением nofollow.

Фильтр uriize нормально работает только с обычным текстом. При попытке обработать им HTML-код результат окажется непредсказуемым;

□ uriizetrunc: <длина>— то же самое, что и uriize, но дополнительно обрезает текст гиперссылок до указанной длины, помещая в его конец символ многоточия:

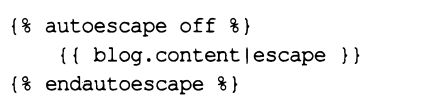
□ safe — подавляет у выводимого значения автоматическое преобразование недопустимых знаков HTML в соответствующие специальные символы;

□ safeseq— подавляет у всех элементов выводимой последовательности автоматическое преобразование недопустимых знаков HTML в соответствующие специальные символы. Обычно применяется совместно с другими фильтрами.

Пример:



□ escape— преобразует недопустимые знаки HTML в соответствующие специальные символы. Обычно применяется в содержимом парного тега autoescape с отключенным автоматическим преобразованием недопустимых знаков. Пример:



□ force escape — то же самое, что и escape, но выполняет преобразование принудительно. Может быть полезен, если требуется провести преобразование у результата, возвращенного другим фильтром;

□ escapejs— преобразует выводимое значение таким образом, чтобы его можно было использовать как строковое значение javascript;

□ striptags —удаляет из выводимого строкового значения все HTML-теги;

□ uriencode— кодирует выводимое значение таким образом, чтобы его можно было включить в состав интернет-адреса (например, передать с GET-napaметром);

□ iriencode— кодирует выводимый интернационализированный идентификатор ресурса (IRI) таким образом, чтобы его можно было включить в состав интернет-адреса (например, передать с GET-параметром);

□ addslashes — добавляет символы обратного слеша перед одинарными и двойными кавычками;

О lj ust: <ширина пространства в символах>—помещает выводимое значение в левой части пространства указанной ширины



Результатом станет строка: "Python ";

□ center: <ширина пространства в символах>— помещает выводимое значение в середине пространства указанной ширины



Результатом станет строка: " Python ";

□ гjust: <ширина пространства в символах>— помещает выводимое значение в правой части пространства указанной ширины



Результатом станет строка: " python";

□ get\_digit :<позиция цифры>— возвращает цифру, присутствующую в выводимом числовом значении по указанной позиции, отсчитываемой справа. Если текущее значение не является числом или если указанная позиция меньше 1 или больше общего количества цифр в числе, то возвращается значение позиции. Пример:

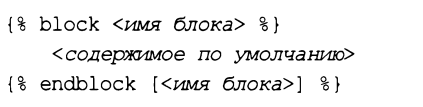


□ linenumbers — выводит строковое значение, разбитое на отдельные строки посредством символов перевода строки, с номерами строк, поставленными слева.

1. Наследование шаблонов

Аналогично наследованию классов в Python, Django предлагает механизм наследования шаблонов. Базовый шаблон содержит элементы, присутствующие на всех страницах сайта: шапку, поддон, главную панель навигации, элементы разметки и др. А производный шаблон выводит лишь уникальное содержимое генерируемой им страницы: список объявлений, объявление, выбранное посетителем, и др.

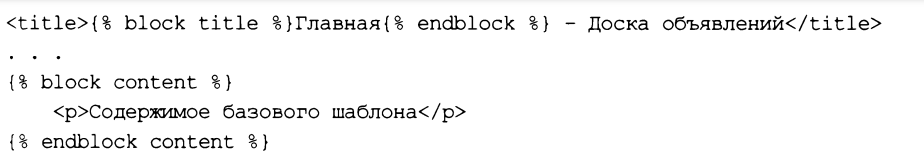
Базовый шаблон содержит блоки, помечающие места, куда будут выведены фрагменты уникального содержимого, сгенерированного производным шаблоном. Блоки в базовом шаблоне объявляются с применением парного тега block . . .Endblock:

****

Имя блока должно быть уникальным в пределах базового шаблона. Его также можно указать в теге endblock, чтобы в дальнейшем не гадать, какому тегу block он соответствует.

Содержимое по умолчанию будет выведено, если производный шаблон его не сгенерирует.

Пример:

****

В коде производного шаблона необходимо явно указать, что он является производным от определенного базового шаблона, вставив в начало его кода тег {% extends<путь к базовому шаблону> %}.

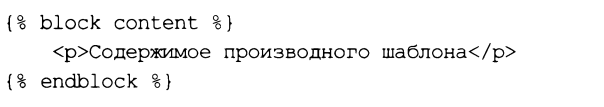
Внимание!

Тег extends должен находиться в самом начале кода шаблона, на отдельной строке.

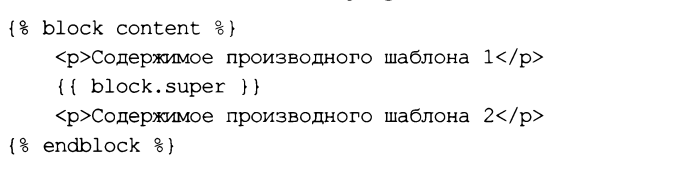
Пример (подразумевается, что базовый шаблон хранится в файле layout\basic.html):



После этого в производном шаблоне точно так же объявляются блоки, но теперь уже в них записывается создаваемое шаблоном содержимое:

****

Содержимое по умолчанию, заданное в базовом шаблоне, в соответствующем блоке производного шаблона доступно через переменную block.super, создаваемую в контексте шаблона самим Django:

****

В результате на экран будут выведены три абзаца:

* Содержимое производного шаблона 1
* Содержимое базового шаблона
* Содержимое производного шаблона 2

В производном шаблоне можно создать другие блоки и таким образом сделать его базовым для других производных шаблонов.

1. Обработка статических файлов

В терминологии Django статическими называются файлы, отправляемые клиенту как есть: таблицы стилей, графические изображения, аудио- и видеоролики, файлы статических веб-страниц и т. п.

Обработку статических файлов выполняет подсистема, реализованная во встроенном приложении django.contrib.staticfiles. Оно включается в список зарегистрированных приложений уже при создании нового проекта, и, если сайт содержит статические файлы, удалять его оттуда нельзя.

Настройка подсистемы статических файлов подсистемой статических файлов управляет ряд настроек, записываемых в модуле Settings.py пакета конфигурации:

□ static url— префикс, добавляемый к интернет-пути статического файла.

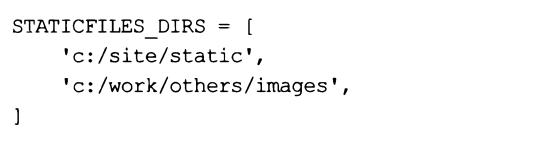
Встретив в начале полученного в запросе пути этот префикс, Django “ поймет”, что запрашивается статический файл. Значение по умолчанию — None, но при создании нового проекта оно устанавливается в ’’/static/";

□ static root — файловый путь к основной папке, в которой хранятся все статические файлы (значение по умолчанию — None).

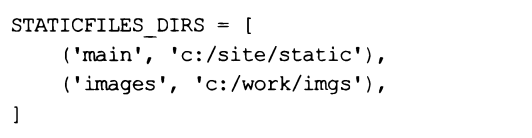
В этой же папке будут собираться все статические файлы, если отдать команду Collectstatic утилиты manage.py;

□ staticfiles dirs — список файловых путей к дополнительным папкам, в которых хранятся статические файлы. Каждый путь может быть задан в двух форматах:

• как строка с файловым путем. Пример:

****

• как кортеж из двух элементов: префикса интернет-пути и файлового пути к папке. Чтобы сослаться на файл, хранящийся в определенной папке, нужно предварить интернет-путь этого файла заданным для папки префиксом. Пример:

****

Теперь, чтобы вывести на страницу файл logo.png, хранящийся в папке с:\work\imgs\others\, следует записать в шаблоне тег:



□ staticfiles finders — список имен классов, реализующих подсистемы поиска статических файлов. По умолчанию включает два класса, объявленные в модуле Django.contrib.staticfiles.finders:

• filesystemfinder— ищет статические файлы в папках, заданных параметрами STATIC\_ROOT И STATICFILES\_DIRS;

• appdirectoriesfinder —ищет статические файлы в папках static, находящихся в пакетах приложений.

Если статические файлы хранятся в каком-то определенном местоположении (только в папках, заданных параметрами static root и staticfiles dirs, или только в папках static в пакетах приложений), можно указать в параметре Staticfiles finders только один класс— соответствующий случаю. Это несколько уменьшит потребление системных ресурсов;

□ staticfiles storage — имя класса, реализующего хранилище статических файлов. По умолчанию используется Хранилище staticfilesstorage из модуля Django.contrib.staticfiles.storage.

Обслуживание статических файлов встроенный в Django отладочный веб-сервер обслуживает статические файлы самостоятельно. Но если сайт находится в эксплуатационном режиме, придется позаботиться об обслуживании статических файлов веб-сервером самостоятельно.

Формирование интернет-адресов статических файлов формировать адреса статических файлов в коде шаблонов можно посредством трех разных программных механизмов:

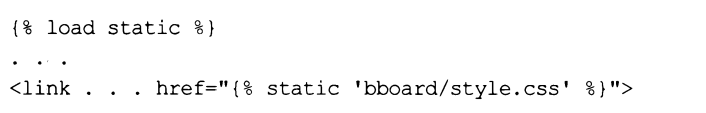
□ static — тег шаблонизатора, вставляющий в шаблон полностью сформированный интернет-адрес статического файла. Формат записи:



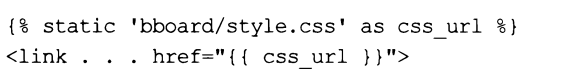
Относительный путь к статическому файлу записывается в виде строки и отсчитывается от папки, путь которой записан в параметрах static root и Staticfiles dirs, или папки static пакета приложения.

Тег реализован в библиотеке тегов с псевдонимом static, которую следует предварительно загрузить тегом load.

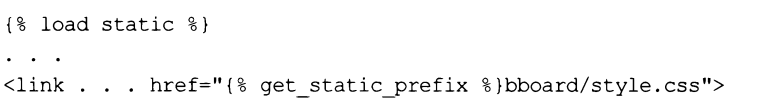
Пример:



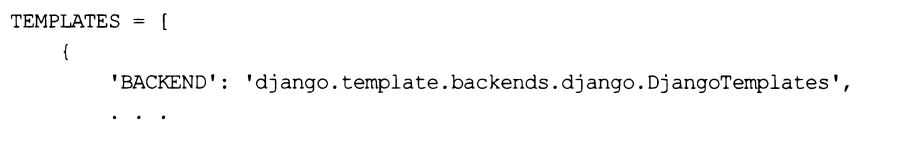
По умолчанию сформированный адрес непосредственно вставляется в код страницы. Также можно сохранить адрес в переменной, записав ее после ключевого слова as. Пример:

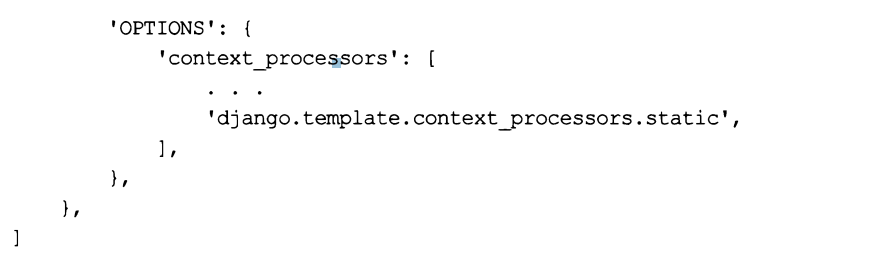
****

□ {% get static prefix %} — тег шаблонизатора, который вставляет в код страницы префикс из параметра static url. Также реализован в библиотеке тегов Static. Пример:



□ django.template.context\_processors.static — обработчик контекста, добавляющий в контекст шаблона переменную static url, которая хранит префикс из одноименного параметра проекта. Поскольку этот обработчик по умолчанию не включен в список активных (элемент context processors параметра options), его нужно добавить туда:





После этого можно обращаться к переменной static url, созданной данным обработчиком в контексте шаблона:

