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.

11.4. 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:

```
{{ bb.published|date:"d.m.Y H:i:s" }}
```

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

```
{{ bb.content|lower|default:'--описания нет--' }}
```

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;
  - n — номер месяца без начального нуля;
  - F и N — полное название месяца в именительном падеже с большой буквы;
  - E — полное название месяца в родительном падеже и в нижнем регистре;
  - M — сокращенное название месяца с большой буквы;
  - b — сокращенное название месяца в нижнем регистре;
  - Y и o — год из четырех цифр;
  - y — год из двух цифр;
  - L — True, если это високосный год, False, если обычный;
  - w — номер дня недели от 0 (воскресенье) до 6 (суббота);
  - l — 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;

- `H` — 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;
- `u` — микросекунды;
- `a` — обозначение половины суток в нижнем регистре (`"д.п."` или `"п.п."`);
- `A` — обозначение половины суток в верхнем регистре (`"ДП"` или `"ПП"`);
- `I` — 1, если сейчас летнее время, 0, если зимнее;
- `P` — 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);
- `e` — название временной зоны;
- `O` — разница между текущим и гринвичским временем в часах;
- `Z` — разница между текущим и гринвичским временем в секундах;
- `c` — дата и время в формате ISO 8601;
- `r` — дата и время в формате RFC 5322;
- `U` — время в формате UNIX (выражается как количество секунд, прошедших с полуночи 1 января 1970 года);
- `T` — название временной зоны, установленной в настройках компьютера.

Пример:

```
{{ bb.published|date:'d.m.Y H:i:s' }}
```

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:

```
{{ bb.published|date:'DATETIME_FORMAT' }}
```

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

```
{{ bb.published|time:'H:i' }}
```

Для вывода времени с применением формата по умолчанию следует использовать обозначение `TIME_FORMAT`:

```
{{ bb.published|time:'TIME_FORMAT' }}
```

или вообще не указывать формат:

```
{{ bb.published|time }}
```

- `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* `False`*>*[*,<строка для* `None`*>*]. Если *строка для* `None` не указана, то вместо нее будет выводиться *строка для* `False` (поскольку `None` будет неявно преобразовываться в `False`).

Примеры:

```
{{ True|yesno }}, {{ False|yesno }}, {{ None|yesno }}
{# Результат: да, нет, может быть #}

{{ True|yesno:'так точно,никак нет,дело темное' }},
{{ False|yesno:'так точно,никак нет,дело темное' }},
{{ None|yesno:'так точно,никак нет,дело темное' }}
{# Результат: так точно, никак нет, дело темное #}

{{ True|yesno:'да,нет' }}, {{ False|yesno:'да,нет' }},
{{ None|yesno:'да,нет' }}
{# Результат: да, нет, нет #}
```

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

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

```
{{ bb.price|default:'У товара нет цены' }}
```

- `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 (...);
- `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 (...);
- `truncatewords_html:<количество слов>` — то же самое, что и `truncatewords`, но сохраняет все HTML-теги, которые встретятся в выводимом значении;
- `wordwrap:<величина>` — выполняет перенос выводимого строкового значения по словам таким образом, чтобы длина каждой получившейся в результате строки не превышала указанную *величину*;
- `cut:<удаляемая подстрока>` — удаляет из выводимого значения все вхождения заданной *подстроки*:

```
{{ 'Python'|cut:'t' }}          {# Результат: 'Pyhon' #}
{{ 'Python'|cut:'th' }}         {# Результат: 'Pyon' #}
```

- `slugify` — преобразует выводимое строковое значение в слаг;
- `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:

```
{{ 34.23234|floatformat }}          {# Результат: 34.2 #}
{{ 34.00000|floatformat }}          {# Результат: 34 #}
{{ 34.26000|floatformat }}          {# Результат: 34.3 #}
{{ 34.23234|floatformat:3 }}        {# Результат: 34.232 #}
{{ 34.00000|floatformat:3 }}        {# Результат: 34.000 #}
{{ 34.26000|floatformat:3 }}        {# Результат: 34.260 #}
```

```
{{ 34.23234|floatformat:-3 }}        {# Результат: 34.232 #}
{{ 34.00000|floatformat:-3 }}        {# Результат: 34 #}
{{ 34.26000|floatformat:-3 }}        {# Результат: 34.260 #}
```

- `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;
- `divisibleby:<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;
- `length_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:

  ```
  {% for bb in bbs|dictsort:'price' %}
      . . .
  {% endfor %}
  ```

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

  ```
  {% for el in list_of_lists|dictsort:1 %}
      . . .
  {% endfor %}
  ```