# Django模板过滤器用法详解

过滤器从字面的意思上，可以理解为：过滤掉不需要的，剩下我们需要的，Django 的模板语言同样也内置了过滤器，如果你了解其他的框架对这个词一定不陌生，比如说 Flask 框架、Vue 框架等，都内置了过滤器这个功能，在本节我们将一起学习 Django 框架的过滤器。

## 1. 过滤器语法格式

过滤器作用是在变量输出时，对输出的变量值做进一步的处理，比如，我们可以使用过滤器来更改变量的输出显示。

过滤器跟模板标签一样，也是在模板中对函数进行调用，比如，对输出的日期进行格式化处理，或者转换大小写字母等，这些都有对应的过滤器去处理它们。当内置过滤器满足不了需求的情况下，也可自定义过滤器。过滤器的语法格式如下：

{{ 变量 | 过滤器1:参数值1 | 过滤器2:参数值2 ... }}
从语法格式我们可以得知过滤器使用|管道符进行变量与过滤器之间的连接，过滤器的可以通过组合多个过滤器实现链式调用，目前过滤器最多接受一个参数。经常使用的过滤器如下表所示：

常见的模板过滤器

| 过滤器 | 使用说明。 |
| --- | --- |
| length | 获取变量的长度，适用于字符串和列表。 |
| lower/upper | 转换字符串为小写/大写形式。 |
| first/last | 获取变量的首个/末尾元素。 |
| add:'n' | 给变量值增加 n。 |
| safe | 默认不对变量内的字符串进行html转义。 |
| cut | 从给定的字符串中删除指定的值。 |
| dictsort | 获取字典列表，并返回按参数中给定键排序的列表。 |
| join | 用字符串连接列表，例如 Python 的 str.join(list)。 |
| truncatewords | 如果字符串字符多于指定的字符数量，那么会被截断。 截断的字符串将以可翻译的省略号序列（“...”）结尾。 |

## 2. 过滤器的实际应用

过滤器相比模板标签要简单的多，我们可以把它们理解成一个 Python 函数，传递参数给他处理就可以了，当滤器接收参数后对它进行处理，最终将处理结果返回到模板中，这就是整个过滤器的实现流程，下面我们通过一些具体的实例，来更加详细理解它的使用方法。

**1)获取变量的长度**

我们使用 length 过滤器得到变量的长度，为了方便演示我们使用 Django shell 编写代码，如下所示：

```
In [1]: from django.template import Template,Context

In [2]: t=Template("""
   ...: <p>hello:{{world|length}}</p>
   ...: """)
   ...: t.render(Context({'world':'you'}))
Out[2]: '¥n<p>hello:3</p>¥n'
```

Django shell 是 Django 提供的交互模式，可以在交互模式下使用项目工程的代码执行相应的操作，从而代替编写视图函数，这样便于测试操作，同时也可以使用它来进行 model 数据表的增删改查，但不能进行过于复杂的操作，它的启动命令是 python manage.py shell。

模板变量 world 使用管道符连接 length 过滤器，最终得到了变量对应值 you 的字符串长度为 3。同时 world 变量的值也可以是列表或者字典，是列表那么将返回列表长度，是字典将返回字典 key 的个数，如若没有定义变量则返回 0。

**2) truncatewords 截取指定个数的词**

在一定数量的单词后截断字符串，语法格式如下所示：

{{ value|truncatewords:2 }}

参数是要截取的字符数量，此处为 2，如果 value 是 "Django is website" ，则输出为"Django is …"，实例代码如下：

```
In [1]: from django.template import Template,Context
In [2]: t=Template("""
    ...: <p>hello:{{value|truncatewords:2}}</p>
    ...: """)
    ...: t.render(Context({'value':'Django is website'}))
Out[2]: '¥n<p>hello:Django is …</p>¥n'
```

**3) dictsort返回指定键的排序列表**

dictsort 它指定字典的键为参数，最后返回按照指定键排序的列表，它的用法如下所示：

```
In [1]: from django.template import Template,Context
In [2]: t=Template("""
   ...: <p>hello:{{value|dictsort:"num"}}
   ...: """)
   ...: t.render(Context({'value':[
   ...:     {'name': 'C语言中文网', 'num':2 },
   ...:     {'name': 'Django官网', 'num': 1},
   ...:     {'name': 'Python官网', 'num': 3},
   ...: ]}))
Out[2]: '¥n<p>hello:[{&#39;name&#39;: &#39;C语言中文网&#39;, &#39;num&#39;: 1}, {&#39;name&#39;: &#39;Django官网&#39;, &#39;num&#39;: 2}, {&#39;name&#39;: &#39;Python官网&#39;, &#39;num&#39;: 3}]¥n'
```

从输出的结果可以看出 dictsort 过滤器对指定的键 num 做了排序处理。当然过滤器也可以与模板标签配合使用，这种属于综合的使用方法，实例如下：

```
In [1]: from django.template import Template,Context
In [2]: t=Template("""
    ...: {% for book in books|dictsort:"author.age" %}
    ...:     {{ book.title }} ({{ book.author.name }})
    ...: {% endfor %}
    ...: """)
    ...: t.render(Context({'books':[
    ...:     {'title': 'C语言教程', 'author': {'name': 'ycs', 'age': 14}},
    ...:     {'title': 'Python教程', 'author': {'name': 'xxw', 'age': 17}},
    ...:     {'title': 'Django教程', 'author': {'name': 'ccs', 'age': 16}},
```

```
    ...: ]}))
Out[2]: '¥n¥n    C语言教程 (ycs)¥n¥n    Django教程 (ccs)¥n¥n    Python教程 (xxw)¥n¥n'
```

提示：&#39 是单引号 HTML 转义后的十进制格式，39 是单引号的 ASCII 编码值。使用 Context 方法的 autoescape=False 参数可以关闭自动转义。

#### 4) add给变量值加 'n'

add 过滤的用法也非常的简单，变量值是整形而且参数也是整形，此时的 add 过滤器相当于加法运算，但是如果变量值和参数都是列表又会怎么样呢，让我们通过下面的例子来看一下：

```
In [1]: from django.template import Template,Context
#整形相加强制转换
In [2]: t=Template("""
    ...: {{value|add:2 }}
    ...: """)
    ...: t.render(Context({'value':'5'}))
Out[2]: '¥n7¥n'

#列表相加
In [3]: t=Template("""
    ...: {{value|add:list}}
    ...: """)
    ...: t.render(Context({'value':['python','Django','Flask'],'list':['Tonado','celery']}))
Out[3]: '¥n[&#39;python&#39;, &#39;Django&#39;, &#39;Flask&#39;, &#39;Tonado&#39;, &#39;celery&#39;]¥n'
```

add 过滤器将首先尝试将两个值都强制转换为整数。如果失败，它将尝试将所有值加在一起。这将对某些数据类型（如字符串，列表等）起作用，而对其他数据类型则失败。如果失败，结果将为空字符串。

### 3. 实现自定义过滤器

Django 在提供诸多可供选择的过滤器的同时，也向开发者提供了自定义过滤器功能，当内置过滤器无法满足开发者的需求的时候，就可以使用自定义过滤器，实现起来也非常的方便。因为自定义过滤器与自定义模板标签，它们的实现过程非常相似，所以我们会在后续章节中做针对性的讲解。

从上述列举的几个实例来看，过滤器的使用方法非常简单，但是它的用处却非常得大，在很大程度上简捷了变量的处理过程，Django 模板内置了很多的过滤器大概有 60 余种，但是有些过滤器不是经常的使用，大家如果有兴趣了解也可以参见 Django 官方（https://docs.djangoproject.com/en/2.2/ref/templates/builtins/）。我们应该尽可能的熟悉它们，俗话说“技多不压身”，这样我们在使用它们时候才能够游刃有余。