1.过滤器简介

过滤器的本质就是函数。
有时候我们不仅仅只是需要输出变量的值，我们还需要修改变量的显示，甚至格式化、运算等等，这就用到了过滤器。

过滤器的使用方式为：变量名 | 过滤器。

2.常用过滤器

常用过滤器如下：
字符串操作：

过滤器	作用	示例
safe	禁用转义	{{ '*hello*' | safe }}

过滤器	作用	示例
capitalize	把变量值的首字母转成大写，其余字母转小写	{ { 'hello' | capitalize }}
lower	字符串转成小写	{ { 'hello' | lower } }
upper	字符串转成大写	{ { 'hello' | lower } }
title	把值中的每个单词的首字母都转成大写	{ { 'hello world' | title }}
trim	把值的首尾空格去掉	{ { ' hello world ' | trim } }
reverse	字符串反转	{ { 'hello' | reverse } }

过滤器	作用	示例
format	格式化输出	{{ '%s is %d' | format('name',17) }}
striptags	渲染之前把值中所有的HTML标签都删掉	{{ 'hello' | striptags }}
truncate	截取前几个字符，后面使用....表示，第一个参数是截取几个字符，第二个参数是是否使用...显示	{{ name | truncate(3,True) }}
escape	开启转义	{{ 'name' | escape }}

列表操作：

过滤器	作用	示例

过滤器	作用	示例
first	取第一个元素	{ { [1,2,3,4,5,6] | first } }
last	取最后一个元素	{ { [1,2,3,4,5,6] | last } }
length	获取列表长度	{ { [1,2,3,4,5,6] | length } }
sum	列表求和	{ { [1,2,3,4,5,6] | sum } }
sort	列表排序	{ { [6,2,3,1,5,4] | sort } }

数值操作：

过滤器	作用	示例
round	四舍五入取整	{ { 12.8888 | round } } 得到13.0
round	向下截取到小数点后2位，返回12.88	{ { 12.8888 | round(2, 'floor') } }得到13.89
abs	绝对值，返回12	{ { -12 | abs } } 得到12

```
https://jinja.palletsprojects.com/en/master/templates/#builtin-filters
```

3.自定义过滤器

当模板内置的过滤器不能满足需求，可以自定义过滤器。

自定义过滤器有两种实现方式：

第一种是通过Flask应用对象的add_template_filter方法。

第二种是通过 template_filter 装饰器来实现自定义过滤器。

自定义过滤器不能和内置过滤器重名，否则会将内置的过滤器覆盖掉。

方法一：

```
def filter_mod_1(num):
    return num % 2

# 该方法第一个参数是函数名，第二个参数是自定义的过滤器名称。
app.add_template_filter(filter_mod_1, 'my_mod_1')
```

方法二：

```
# 通过装饰器，自定义过滤器，装饰器接受一个参数：过滤器名称
# 过滤器函数接收变量
@app.template_filter('my_mod')
def filter_mod(num):
    return num % 2
```

带参数过滤器

```
@app.template_filter('filter_list')
def filter_list(li, start, end):
    """自定义过滤器，自定义截取列表区间"""
    return li[start:end]
```

在模板中调用传参数过滤器后面加小括号，就跟python调用方法一样：

```
{% raw %}
    {{a|my_mod}} {#没参数#}
    {{b|filter_list(2,5)}} {# 有参数#}
{% endraw %}
```

注意

有时想要或甚至必要让 Jinja 忽略部分，不会把它作为变量或块来处理。

例如，如果 使用默认语法，你想在在使用把 {{ 作为原始字符串使用，并且不会开始一个变量 的语法结构，你需要使用一个技巧。

最简单的方法是在变量分隔符中（ {{ ）使用变量表达式输出：

```
{{ '{{' }}
```

对于较大的段落，标记一个块为 raw 是有意义的。例如展示 Jinja 语法的实例， 你可以在模板中用这个片段:

```
{% raw %}
    <ul>
    {% for item in seq %}
        <li>{{ item }}</li>
    {% endfor %}
    </ul>
{% endraw %}
```