**https://blog.csdn.net/plan\_jok/article/details/109725095**

**render\_template**

* + [render\_template简介](https://blog.csdn.net/plan_jok/article/details/109725095#render_template_1)
  + [模板语法，获取变量](https://blog.csdn.net/plan_jok/article/details/109725095#_29)
  + [模板语法，分支循环判断](https://blog.csdn.net/plan_jok/article/details/109725095#_85)
  + [系统字符串过滤器](https://blog.csdn.net/plan_jok/article/details/109725095#_167)
  + [系统列表过滤器](https://blog.csdn.net/plan_jok/article/details/109725095#_242)
  + [自定义过滤器](https://blog.csdn.net/plan_jok/article/details/109725095#_309)
    - [案例1：获取列表的偶数和 案例2：反转列表](https://blog.csdn.net/plan_jok/article/details/109725095#1____2_321)

**render\_template简介**

* 解释：属于jinja2的模板函数
* 好处：
  + 以后的视图函数，只负责业务逻辑的处理，比如：数据库的增删改查
  + 以后数据的展示，全部都由jinja2的模板负责
* 使用格式
* response = render\_template('模板文件')，模板文件一般存储在templates目录下

示例：

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

response = render\_template('file01.html')

return response

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**模板语法，获取变量**

* 解释：在模板中获取视图函数的变量
* 格式：
* {{变量}}

示例：  
python代码：

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

number = 10

str = 'lili'

tuple = (1, 2, 3, 4, 5, 6)

list = [7, 8, 9, 10, 11, 12]

dict = {

'name': "lili",

'age': 18

}

return render\_template("file02variable.html", number=number, str=str, tuple=tuple, list=list, dict=dict)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

页面代码：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<h1>1.获取各种变量的值</h1>

<h2>整数：{{number}}</h2>

<h2>字符串：{{str}}</h2>

<h2>元祖：{{tuple}}，分开获取：{{tuple[0]}}，{{tuple.1}}</h2>

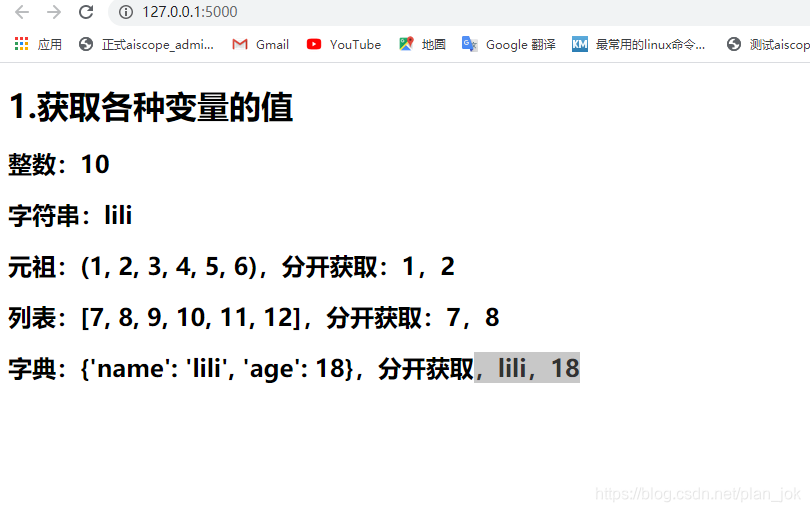
<h2>列表：{{list}}，分开获取：{{list.0}}，{{list.1}}</h2>

<!--如果字典使用方括号，获取，需要写成字符串，如果不是字符串，那么则会被当成变量对待-->

<h2>字典：{{dict}}，分开获取，{{dict.name}}，{{dict["age"]}}</h2>

</body>

</html>

结果展示：  


**模板语法，分支循环判断**

* 模板语法的种类：
  + 分支格式：
  + {% if 条件 %}
  + 语句1
  + {% elif %}
  + 语句2
  + {% else % }
  + 语句3
  + {% endif %}
  + 循环语句格式：
  + {% for 变量 in 容器 %}
  + 语句
  + {% endfor %}
  + 注释：
  + {# 这里是注释的内容 #}

示例：  
python代码：

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

number = 10

str = 'lili'

tuple = (1, 2, 3, 4, 5, 6)

list = [7, 8, 9, 10, 11, 12]

dict = {

'name': "lili",

'age': 18

}

return render\_template("file03\_other\_programer.html", number=number, str=str, tuple=tuple, list=list, dict=dict)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

页面代码：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<h1>1.遍历元祖中的偶数</h1>

{% for item in tuple %}

{% if item %2 == 0 %}

{{ item }}

{% endif %}

{% endfor %}

<h1>2.遍历字典</h1>

{% for key in dict %}

<h3>

{# dict.key那么这个key会被当成字典中的一个键，dict[key],那么这个key当成一个变量 #}

{{ key }} = {{ dict[key] }}

</h3>

{% endfor %}

</body>

</html>

结果展示：  


**系统字符串过滤器**

* 解释：过滤器，用来过滤想要的数据
* 格式：
* {{ 字符串 | 字符串过滤器 }}
* 常见的字符串过滤器：
* safe:禁用转移，让标签生效
* capitalize:把变量值得首字母转成大写，其余字母小写
* lower:把值转成小写
* upper:把值转成大写，中文没有大小写
* title:把值中的每个单词的首字母都转成大写
* reverse:字符串反转
* format:格式化输出
* striptags:渲染之前把值中的所有的HTML标签都删掉

示例：  
python代码：

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template("file04string\_filter.html")

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

页面代码：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

{# 使用格式 {{ 字符串 | 字符串过滤器 }} #}

1.safe:禁用转移，让标签生效

<p>{{ '<em>hello</em>' | safe }}</p>

2.capitalize:把变量值得首字母转成大写，其余字母小写

<p>{{ 'hello PYTHON' | capitalize }}</p>

3.lower:把值转成小写

<p>{{ 'HELLO PyTHon' | lower }}</p>

4.upper:把值转成大写，中文没有大小写

<p>{{ 'hello python 你好' | upper }}</p>

5.title:把值中的每个单词的首字母都转成大写

<p>{{ 'hello python' | title }}</p>

6.reverse:字符串反转

<p>{{ 'hello' | reverse }}</p>

<p>{{ '无敌' | reverse }}</p>

7.format:格式化输出

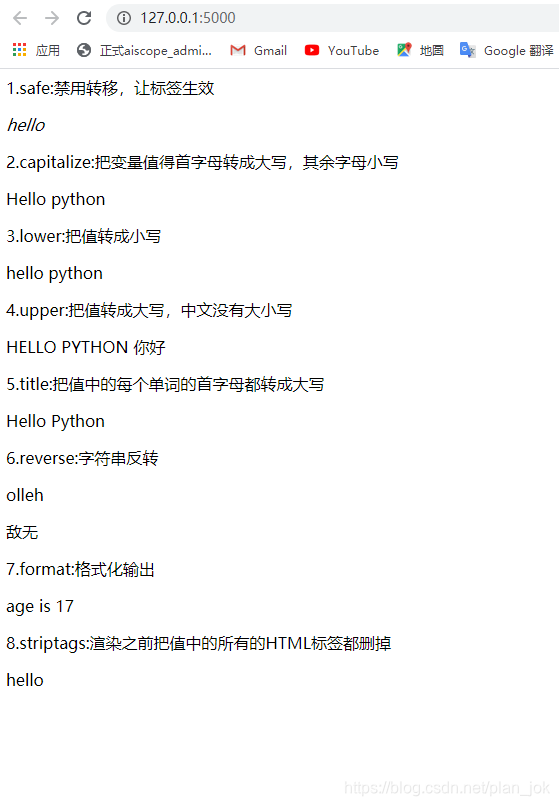
<p>{{ '%s is %s' | format('age',17) }}</p>

8.striptags:渲染之前把值中的所有的HTML标签都删掉

<p>{{ '<em>hello</em>' | striptags }}</p>

</body>

</html>

结果展示：  


**系统列表过滤器**

* 解释：过滤器，用来过滤想要的数据
* 格式：
* {{ 列表 | 列表过滤器 }}
* 常见的列表过滤器：
* first:取第一个元素
* last:取最后一个元素
* length:获取列表长度
* sum:列表求和
* sort:列表排序,默认升序

示例：  
python代码：

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return render\_template("file05list\_filter.html")

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

页面代码：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

{# 使用格式{{ 列表 | 列表过滤器 }} #}

1.first:取第一个元素

<P>{{ [1,2,3,4,5,6,7,8,9] |first }}</P>

2.last:取最后一个元素

<P>{{ [1,2,3,4,5,6,7,8,9] |last }}</P>

3.length:获取列表长度

<P>{{ [1,2,3,4,5,6,7,8,9] |length }}</P>

4.sum:列表求和

<P>{{ [1,2,3,4,5,6,7,8,9] |sum }}</P>

5.sort:列表排序,默认升序

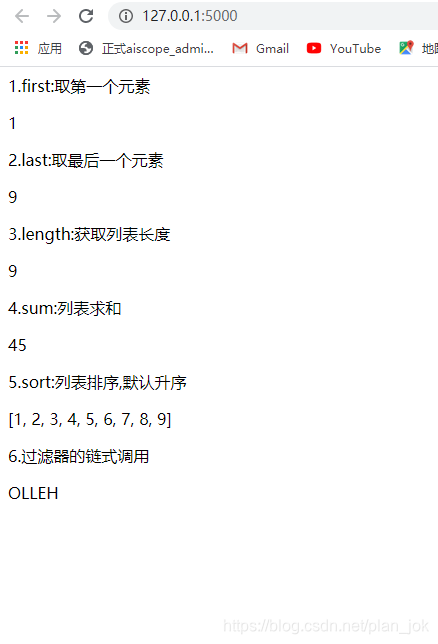
<P>{{ [1,2,3,4,5,6,7,8,9] |sort }}</P>

6.过滤器的链式调用

<p>{{ 'hello' | upper | reverse }}</p>

</body>

</html>

结果展示：  


**自定义过滤器**

* 解释：当系统提供的过滤器满足不了需求的时候，需要自定义
* 自定义过滤器有两种格式：
  + 先定义好函数，再讲函数添加到系统默认的过滤器列表中
  + def 函数名()：pass
  + app.add\_template\_filter(函数名，'过滤器名字')
  + 定义函数的时候，直接使用系统过滤器进行装饰
  + @app.template\_filter('过滤器名称')
  + def 函数名():
  + pass

**案例1：获取列表的偶数和 案例2：反转列表**

python代码：

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

# 案例一

# 1.先定义好函数，再将函数添加到系统默认的过滤器列表中

def get\_even(list):

sum = 0

for i in list:

if i % 2 == 0:

sum += i

return sum

# 参数1：关联的函数名称，参数2：在模板中使用的过滤器名字

app.add\_template\_filter(get\_even, "even")

# 案例二

# 2.定义函数的时候，直接使用系统过滤器进行装饰

@app.template\_filter("reserve")

def list\_reserve(list):

list.reverse()

return list

@app.route('/')

def hello\_world():

return render\_template("file06custom\_filter.html")

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

页面代码：

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<h2>原列表：{{ [1,2,3,4,5,6,7] }}</h2>

<h2>偶数列表：{{ [1,2,3,4,5,6,7] | even }} </h2>

<h2>反转列表：{{ [1,2,3,4,5,6,7] | reserve }} </h2>

<h2>降序列表：{{ [1,2,3,4,5,6,7,10,8,19,44,32,34,12,2,3] | sort | reserve }} </h2>

</body>

</html>

结果展示：  
