# twig文档

目录

[twig文档 1](#_Toc453848852)

[基本概念 2](#_Toc453848853)

[1、tag 3](#_Toc453848854)

[1.1、继承模板 extends 3](#_Toc453848855)

[1.2、模块block 4](#_Toc453848856)

[1.3、for循环 5](#_Toc453848857)

[1.4、if判断 7](#_Toc453848858)

[1.5、macro标签 8](#_Toc453848859)

[1.6、include标签 9](#_Toc453848860)

[1.7、import标签 10](#_Toc453848861)

[1.8、use标签 10](#_Toc453848862)

[1.9、set标签 12](#_Toc453848863)

[1.10、spacelsee标签 13](#_Toc453848864)

[1.11、autoescape标签 13](#_Toc453848865)

[1.12、raw标签 13](#_Toc453848866)

[1.13、flush标签 14](#_Toc453848867)

[1.14、do标签 14](#_Toc453848868)

[2、filter 过滤器 15](#_Toc453848869)

[2.1 data 15](#_Toc453848870)

[2.2 format 15](#_Toc453848871)

[2.3 replace 15](#_Toc453848872)

[2.4 number\_format 16](#_Toc453848873)

[2.5 url\_encode 16](#_Toc453848874)

[2.6 json\_encode 16](#_Toc453848875)

[2.7 convert\_encoding 16](#_Toc453848876)

[2.8 title 17](#_Toc453848877)

[2.9 capitalize 17](#_Toc453848878)

[2.10 nl2br 17](#_Toc453848879)

[2.11 upper 18](#_Toc453848880)

[2.12 lower 18](#_Toc453848881)

[2.13 striptags 18](#_Toc453848882)

[2.14 join 19](#_Toc453848883)

[2.15 reverse 19](#_Toc453848884)

[2.16 length 19](#_Toc453848885)

[2.17 sort 20](#_Toc453848886)

[2.18 default 20](#_Toc453848887)

[2.19 keys 20](#_Toc453848888)

[2.20 escape 21](#_Toc453848889)

[2.21 raw 21](#_Toc453848890)

[2.22 merge 21](#_Toc453848891)

[3、function 函数 21](#_Toc453848892)

[3.1 attribute 22](#_Toc453848893)

[3.2 block 22](#_Toc453848894)

[3.3 constant 22](#_Toc453848895)

[3.4 cycle 22](#_Toc453848896)

[3.5 dump 23](#_Toc453848897)

[3..6 parent 23](#_Toc453848898)

[3.7 random 24](#_Toc453848899)

[3.8 range 24](#_Toc453848900)

[4、检测 25](#_Toc453848901)

[4.1 divisibleby 25](#_Toc453848902)

[4.2 even 25](#_Toc453848903)

[4.3 odd 25](#_Toc453848904)

## 基本概念

输出内容

{{ 内容 }}

自带的全局变量

\_self  这个参看macro标签

\_context 这个就是当前的环境

\_charset: 当前的字符编码

变量赋值

{% set variable = value %}

{{ foo.bar }} === {{ foo['bar'] }}

使用-操作符，可以很方便的删除TWIG标签之前或之后与html标签之间的空白。

{% set value = 'no spaces' %}

{#- No leading/trailing whitespace -#}

{%- if true -%}

    {{- value -}}

{%- endif -%}

{# output 'no spaces' #}

## 1、tag

### 1.1、继承模板 extends

这个标签用来表示本模板继承自另外一个模板和[**PHP**](http://lib.csdn.net/base/36)一样，twig不支持多重继承，所以你只能有一个extends标签，而且要在模板的最上方。

动态继承

{% extends some\_var %}

如果变量是一个twig模板对象，也可以。

$layout = $twig->loadTemplate('some\_layout\_template.twig');

$twig->display('template.twig', array('layout' => $layout));

1.2版本更新 你可以传递一个数组，twig会选择第一个存在的模板，来继承。

{% extends ['layout.html', 'base\_layout.html'] %}

条件继承

{% extends standalone ? "minimum.html" : "base.html" %}

### 1.2、模块block

一级模块只能跟继承的父模板中包含的block，

在block中可自行创建其他的block（支持嵌套，但不支持同名）

继承父模板中的block中的内容，也即是说是添加，而不是覆盖，可添加方法{{ parent() }}

### 1.3、for循环

基于数组的循环

{% for user in users %}

      <li>{{ user.username|e }}</li>

  {% endfor %}

基于数字的循环，特别要注意，这里会输出0-10 也就是11个数字。

{% for i in 0..10 %}

    \* {{ i }}

{% endfor %}

基于字母的循环

{% for letter in 'a'..'z' %}

    \* {{ letter }}

{% endfor %}

在循环体内部的变量

| **变量名** | **描述** |
| --- | --- |
| loop.index | 循环的次数（从1开始） |
| loop.index0 | 循环的次数（从0开始） |
| loop.revindex | 循环剩余次数（最小值为1） |
| loop.revindex0 | 循环剩余次数（最小值为0） |
| loop.first | 当第一次循环的时候返回true |
| loop.last | 当最后一次循环的时候返回true |
| loop.length | 循环的总数 |
| loop.parent | 被循环的数组 |

添加条件判断循环

{% for user in users if user.active %}

      <li>{{ user.username|e }}</li>

  {% endfor %}

{% esle %}分支

当数组没有数据的时候就会返回else里面的信息

按key循环

{% for key in users|keys %}

        <li>{{ key }}</li>

  {% endfor %}

按key，value循环

{% for key, user in users %}

       <li>{{ key }}: {{ user.username|e }}</li>

  {% endfor %}

### 1.4、if判断

{% if users %}

    <ul>

        {% for user in users %}

            <li>{{ user.username|e }}</li>

        {% endfor %}

    </ul>

{% endif %}

{% if kenny.sick %}

    Kenny is sick.

{% elseif kenny.dead %}

    You killed Kenny!  You bastard!!!

{% else %}

    Kenny looks okay --- so far

{% endif %}

### 1.5、macro标签

macro（宏标签）类似于其他语言中的函数，常用于填充html标签，以下是一个例子，用来渲染<input>

{% macro input(name, value, type, size) %}

    <input type="{{ type|default('text') }}" name="{{ name }}" value="{{ value|e }}" size="{{ size|default(20) }}" />

{% endmacro %}

macro与函数的不同之处在于：

1、参数的默认值是通过macro块内部的 default过滤器来定义的。

2、参数总是可选的。

另外，就跟php函数一样，macro内部是无法使用外部的变量的。但你可以传递一个特殊变量\_context作为参数来获取整个内容。

macro可以被定义在任何的模板内，但在你使用之前需要使用 imported

如果是调用当前模板中定义的macro的话就不需要include引用，（当然，你也可以用include，但是没什么必要），可直接使用\_self来调用

{{ \_self.textarea('comment') }}

如果两个macro在不同的文件里，你需要使用import

{% import "forms.html" as forms %}

还可以重命名

{% from 'forms.html' import input as input\_field, textarea %}

### 1.6、include标签

载入一个模板，返回渲染的内容。载入的模板可以使用当前模板的变量

{% include 'header.html' %}

你可以给模板添加变量

{% include 'foo' with {'foo': 'bar'} %}

你也可以使用 only 关键字 来禁止载入的模板使用当前模板的变量，只能使用include 时with的变量

{% include 'foo' with {'foo': 'bar'} only %}

{% include 'foo' only %}

载入的模板名也可以是一个twig表达式

{% include ajax ? 'ajax.html' : 'not\_ajax.html' %}

{% include some\_var %}

1.2版本新加内容，可以在模板加上 ignore missing 关键字，这样当模板不存在的时候就不会引发错误。

{% include "sidebar.html" ignore missing %}

{% include "sidebar.html" ignore missing with {'foo': 'bar} %}

{% include "sidebar.html" ignore missing only %}

1.2版本新加内容，你可以给include传递一个数组，他会自动载入第一个存在的模板

{% include ['page\_detailed.html', 'page.html'] %}

### 1.7、import标签

参见macro标签

### 1.8、use标签

use标签是1.1版本新添加内容。

这个use标签主要是来解决模板只能从一个父模板继承，而你又想重用其他模板的问题。但是use标签只会导入block区块，

（注意import只会导入宏macros，include会导入一切。这三个标签要区分清楚）

{% use "blocks.html" %}

要注意，被use标签导入的模板(上例中的block.html)，不能再继承别的模板，不能定义宏macros。但它可以再use其他模板。

另外use标签后面的文件名，不能是一个表达式。

当被导入了的block和主模板的block重名了，模板引擎会自动忽略被use标签导入block。

为了避免这种情况。你可以在使用use标签的时候，给block重命名

{% use "blocks.html" with sidebar as base\_sidebar %}

1.3版本新支持了 parent()函数，（这个特别重要）

parent()函数，会自动的搞定block的继承树，如果你在主模板里覆盖了use标签引入进来的block块，而用parent()函数则可以调用被覆盖的那个block内容

如果你在use标签里给导入的block重命名了，那就可以使用block函数，来代替上面代码中的parent函数所达到的效果

{% use "blocks.html" with sidebar as parent\_sidebar %}

{% block sidebar %}

  {{ block('parent\_sidebar') }}

{% endblock %}

你可以使用任意数量的use标签，如果多个use标签里的block名字存在重复，那么最后use的那个有效。

### 1.9、set标签

主要是用来给变量赋值的。

{% set foo = 'foo' %} 简单赋值

{% set foo = [1, 2] %} 定义数组

{% set foo = {'foo': 'bar'} %} 定义key，value数组

{% set foo = 'foo' ~ 'bar' %} 强制拼接字符串

{% set foo, bar = 'foo', 'bar' %} 同事定义多个变量

set还有一种用法，就是把 块内的内容赋值给变量

{% set foo %}

  <div id="pagination">

    ...

  </div>

{% endset %}

### 1.10、spaceless标签

会删除html标签之间的空白

{% spaceless %}

<div>

    <strong>foo</strong>

</div>

{% endspaceless %}

{# output will be <div><strong>foo</strong></div> #}

### 1.11、autoescape标签

### 1.12、raw标签

raw标签，保证区块内的数据不被模板引擎解析，而直接输出，一般文章新闻类的需要这种方式

{% raw %}

    <ul>

    {% for item in seq %}

        <li>{{ item }}</li>

    {% endfor %}

    </ul>

{% endraw %}

方式二：使用条件的形式

{{ data|raw }}

### 1.13、flush标签

{% flush %}

1.5版本新增内容

告诉模板，刷新输出缓存，在内部其实是调用了php的flush函数

### 1.14、do标签

1.5版本新增内容

do 标签的作用就像是输出标签一样{{ }}，他可以计算一些表达式，区别是不打印出任何东西

{% do 1 + 2 %}

### 1.15 render 渲染

直接渲染

{{ render(controller('HouseBundle:News:list',{'id':56})) }}

HouseBundle:News:list 表示一个controller地址 后面的待确定的

保持动态 （例如：登录状态）

{{ render\_esi(controller('HouseBundle:News:list',{'id':56})) }}

## 2、filter 过滤器

### 2.1 data

1.1版本新增时区支持，1.5版本增加了默认的日期格式。

这个过滤器和php的date函数无限类似

{{ post.published\_at|date("m/d/Y") }}

{{ "now"|date("m/d/Y") }}

如果想在格式里输出字母，需要在每个字母前输入\\

{{ post.published\_at|date("F jS \\a\\t g:ia") }}

你可以指定时区

{{ post.published\_at|date("m/d/Y", "Europe/Paris") }}

### 2.2 format

和php的printf函数一样，用来替换占位符

{{ "I like %s and %s."|format(foo, "bar") }}

### 2.3 replace

替换

{{ "I like %this% and %that%."|replace({'%this%': foo, '%that%': "bar"}) }}

### 2.4 number\_format

1.5版本新增过滤器。

他是php函数 number\_format的一个包装 直接见函数参考吧

{{ 200.35|number\_format }} 输出整数（200）

### 2.5 url\_encode

这个直接使用 urlencode函数

{{ data|url\_encode() }}

### 2.6 json\_encode

{{ data|json\_encode() }}

### 2.7 convert\_encoding

1.4版本新加内容

转换一个字符串，第一个参数是输出编码，第二个参数是输入编码

本函数依赖于iconv 或者mbstring 所以至少需要安装一个

{{ data|convert\_encoding('UTF-8', 'iso-2022-jp') }}

### 2.8 title

会让每个单词的首字母大写。

{{ 'my first car'|title }} My First Car

### 2.9 capitalize

会把字符串变成 首字母大写，其余字母小写的格式

{{ 'my first car'|capitalize }}  My first car

### 2.10 nl2br

会把换行符\n 变成<br />

{{ "I like Twig.\nYou will like it too."|nl2br }}

{# outputs

    I like Twig.<br />

    You will like it too.

#}

### 2.11 upper

让字符串变大写

{{ "I like Twig.\nYou will like it too."| upper }}

### 2.12 lower

让字符串变小写

{{ "I like Twig.\nYou will like it too."| lower }}

### 2.13 striptags

html过滤

{{ "<p>I like Twig.\nYou will like it too.</p>"|striptags }}

输出p里面的内容，不包括p

### 2.14 join

跟python的join一样，用来将一个数组的内容连接起来，并用指定的字符串分割。

{{ [1, 2, 3]|join }}

{# returns 123 #}

{{ [1, 2, 3]|join('|') }}

{# returns 1|2|3 #}

### 2.15 reverse

反转一个数组，或者是一个实现了Iterator接口的对象

### 2.16 length

返回数组或者字符串的长度

{% if users|length > 10 %}

    ...

{% endif %}

### 2.17 sort

分类

### 2.18 default

当变量没定义或者为空的时候，返回预先设置的内容

{{ var|default('var is not defined') }}

{{ var.foo|default('foo item on var is not defined') }}

{{ var['foo']|default('foo item on var is not defined') }}

{{ ''|default('passed var is empty')  }}

### 2.19 keys

返回key数组

{% for key in array|keys %}

    ...

{% endfor %}

### 2.20 escape

主要转义  & < > ' " 。并且它有个简写方式 e

{{ user.username|escape }}

{{ user.username|e }}

### 2.21 raw

直接输出内容，不经过转译编码

### 2.22 merge

合并数组

{% set items = { 'apple': 'fruit', 'orange': 'fruit' } %}

{% set items = items|merge({ 'peugeot': 'car' }) %}

{# items now contains { 'apple': 'fruit', 'orange': 'fruit', 'peugeot': 'car' } #}

## 3、function 函数

### 3.1 attribute

1.2版本新增

他就相当于是个. 操作符。

{{ attribute(object, method) }}

### 3.2 block

输出区块的内容

{{ block(‘title’) }} 输出block名称为title的值

### 3.3 constant

读取常量

{{ some\_date|date(constant('DATE\_W3C')) }}

{{ constant('Namespace\\Classname::CONSTANT\_NAME') }}

### 3.4 cycle

循环输出数组的内容

{% set fruits = ['apple', 'orange', 'citrus'] %}

{% for i in 0..10 %}

    {{ cycle(fruits, i) }}

{% endfor %}

{{ cycle(fruits, i) }}

fruits 数组

当i比数组的长度还要大的时候，就会从头开始继续遍历数组

### 3.5 dump

1.5版本新增

打印变量，就是用的php的var\_dump函数，

另外twig默认是没有开启debug模式的，你需要首先开启他

{{ dump(ident.XX) }}

打印数据

### 3..6 parent

获取父block的内容.

{% block sidebar %}

    <h3>Table Of Contents</h3>

    ...

    {{ parent() }}

{% endblock %}

### 3.7 random

1.5版本新增，从一个数组中随机返回一个

{{ random(['apple', 'orange', 'citrus']) }}

### 3.8 range

返回一个数字数组，从第一个参数开始，到第二个参数结束（包含第二个），第三个参数是步长（可以省略）。 和0..10一样

{% for i in range(0, 3) %}

    {{ i }},

{% endfor %}

{# returns 0, 1, 2, 3 #}

## 4、检测

### 4.1 divisibleby

检查是否能被整除

{% if loop.index is divisibleby(3) %}

    ...

{% endif %}

### 4.2 even

变量是否是偶数

{{ var is even }}

### 4.3 odd

变量是否是奇数

## 5、数据调用

### 5.1 获取url上的参数值

{% set id = app.request.get('id') %} 获取url上的id所传回来的值

### 5.2获取某条数据中的一个字段的值

{% set newsDataName = ('house.news')|getOne("name",{id: id}) %}

其中house.news 是表示服务器名

name是需要获取的字段

id是根据某种条件查询到某条数据（此处是根据id来查询某条数据）

### 5.3调用N条数据

{% set categoryData2 = ('house.category')|getAll({pid:item.id}) %}

house.category 表示服务器名

pid表示需要根据某种条件来查询数据

### 5.4调用N条数据中的某一个字段

{% set data = ('house.category')|getCol('name', {pid:0}) %}

house.category 表示服务器名

pid表示需要查询的条件

name 表示需要查询的某个字段

### 5.5 url解析

{{('news/list')|U({category: item.id})}}

('news/list') 控制器

U后面接的是参数，也就是跳转url后面所需要带的参数

### 5.6 路径解析

css、js文件路径解析

<link rel="stylesheet" href="{{ asset('bundles/'~bundlepath~'/css/global.css', absolute=true) }}" />

图片路径解析

<img alt="{{item.name}}" src="{{assetImg(item.thumb,{type:2})}}" height="120" width="180" />