模板

用户通过浏览器访问的目的是为了看到数据嵌在网页中，也就是说给浏览器提供一个网页HTML即可。

网页内容分为：

- 静态内容：存储在服务器端文件系统里，这些文件持久不变。例如，HTML、CSS、JS文件
- 动态内容：通过URL映射到函数，动态生成内容
    - 所有内容字符串，完全由代码生成。由于全部是代码生成，包括其中的HTML部分，实际上就是在做字符串拼接，如果修改HTML，将非常痛苦。显示和数据代码耦合在一起，非常不好
    - 读取一个HTML文件的内容，替换其中部分字符串。管显示的HTML是一个单独的文件（模板），生成数据的代码是一个独立的函数，写在单独的py文件中，只是将HTML内容发给用户前做字符串替换即可。好处是，显示和数据代码分离

生成环境中，会对静态、动态的内容的访问进行区别对待，这就是`动静分离`。

模板配置

参考 https://docs.djangoproject.com/en/3.2/topics/templates/#configuration

在主配置文件settings.py中TEMPLATES

- DIRS：依次搜索模板的目录列表
    - BASE_DIR / 'templates' 表示项目根目录下templates目录
- APP_DIRS：布尔值。告诉引擎是否在 INSTALLED_APPS 中应用目录中搜索模板。
    - True表示**DjangoTemplates**引擎会在每一个应用目录下的templates目录中搜索模板

公共模板路径可以配置在DIRS列表中，应用自己使用的模板定义在应用目录下的templates目录中

```
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

在项目根目录下创建`templates`目录，存放模板文件。

模板渲染

模板页

templates/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>马哥教育Django首页</title>
</head>
<body>
    我是模板页，数据{{content}}
</body>
</html>
```

模板处理

模板处理的2个步骤

1. 加载模板文件，读取文件内容
2. 渲染，替换内容中占位符，返回最终的字符串

render函数

`render(request, template_name, context=None, content_type=None, status=None, using=None)`

- template_name模板名
- context渲染用上下文字典
- status响应报文的状态码
- 返回HttpResponse对象

```
from django.shortcuts import render
from django.http import HttpRequest, HttpResponse

def test_index(request:HttpRequest):
    context = {'test':'测试字符串'}
    content = render(request, 'index.html', context)
    return HttpResponse(content)
```

DTL语法

Django的模板引擎默认使用Django template language (DTL)构建。

1 变量

`语法 {{ variable }}`

变量名由字母、数字、下划线、点号组成。

点号使用的时候，例如foo.bar，遵循以下顺序：

1. 字典查找，例如foo["bar"]，把foo当做字典，bar当做key

2. 属性或方法的查找，例如foo.bar，把foo当做对象，bar当做属性或方法
3. 数字索引查找，例如foo[bar]，把foo当做列表一样，使用索引访问

```
def index(request:HttpRequest):
    """视图函数：请求进来返回响应"""
    mydict = {
        'a':100,
        'b':0,
        'c':list(range(10,20)),
        'd':'abc',
        'date':datetime.datetime.now()
    }
    context = {'content':'www.magedu.com', 'mydict':mydict}
    return render(request, 'index.html', context)
```

如果**变量未能找到，则缺省插入空字符串''**
在模板中调用方法，**不能加小括号**，自然也不能传递参数。
{{ mydict.a }}符合第一条，当做字典的key就可以访问到了
{{ mydict.keys }} 这样是对的，不能写成{{ mydict.keys() }}。符合第二条，当做mydict对象的属性或方法。
{{ mydict.c.0 }} 符合第三条。

2 模板标签

标签采用{% tag %}语法。

if/else 标签
基本语法格式如下：

```
{% if condition %}
     ... display
{% endif %}
```

或者：

```
{% if condition1 %}
   ... display 1
{% elif condition2 %}
   ... display 2
{% else %}
   ... display 3
{% endif %}
```

条件也支持and、or、not
注意，因为这些标签是断开的，所以不能像Python一样使用缩进就可以表示出来，必须有个结束标签，例如endif、endfor。

for 标签
https://docs.djangoproject.com/en/2.0/ref/templates/builtins/#for

```
<ul>
{% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
{% endfor %}
</ul>

{%  for person  in  person_list %}
<li>    {{  person.name }}  </li>
{%  endfor  %}
```

变量	说明
forloop.counter	当前循环从1开始的计数
forloop.counter0	当前循环从0开始的计数
forloop.revcounter	从循环的末尾开始倒计数到1
forloop.revcounter0	从循环的末尾开始到计数到0
forloop.first	第一次进入循环
forloop.last	最后一次进入循环
forloop.parentloop	循环嵌套时，内层当前循环的外层循环

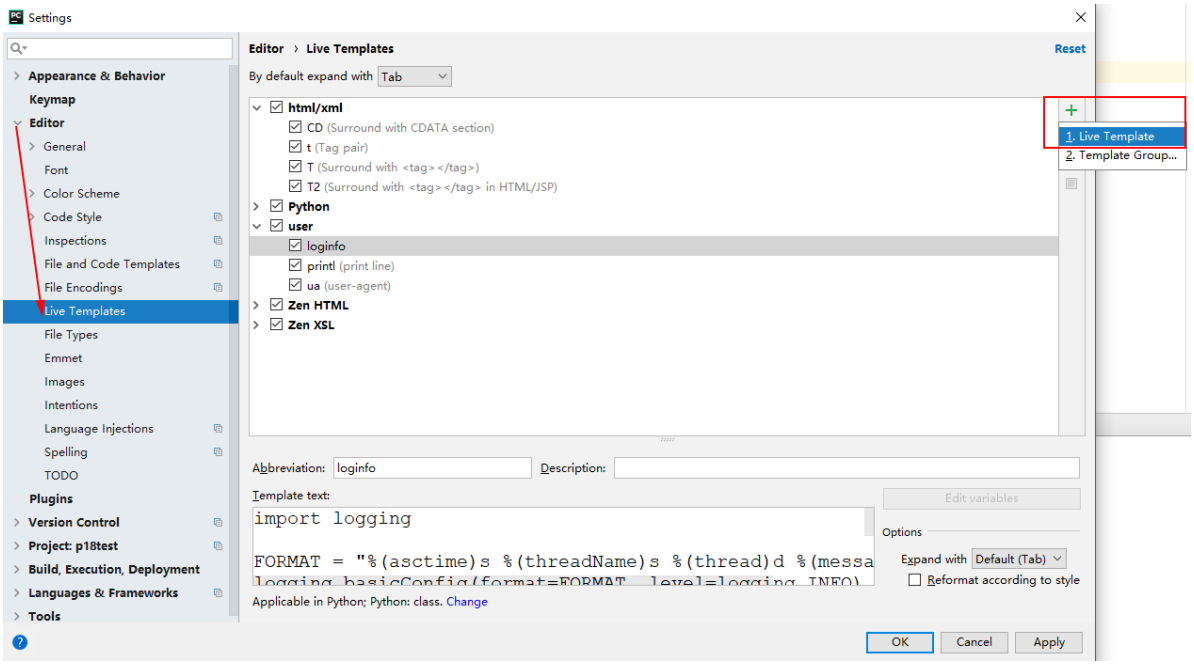
给标签增加一个 reversed 使得该列表被反向迭代：

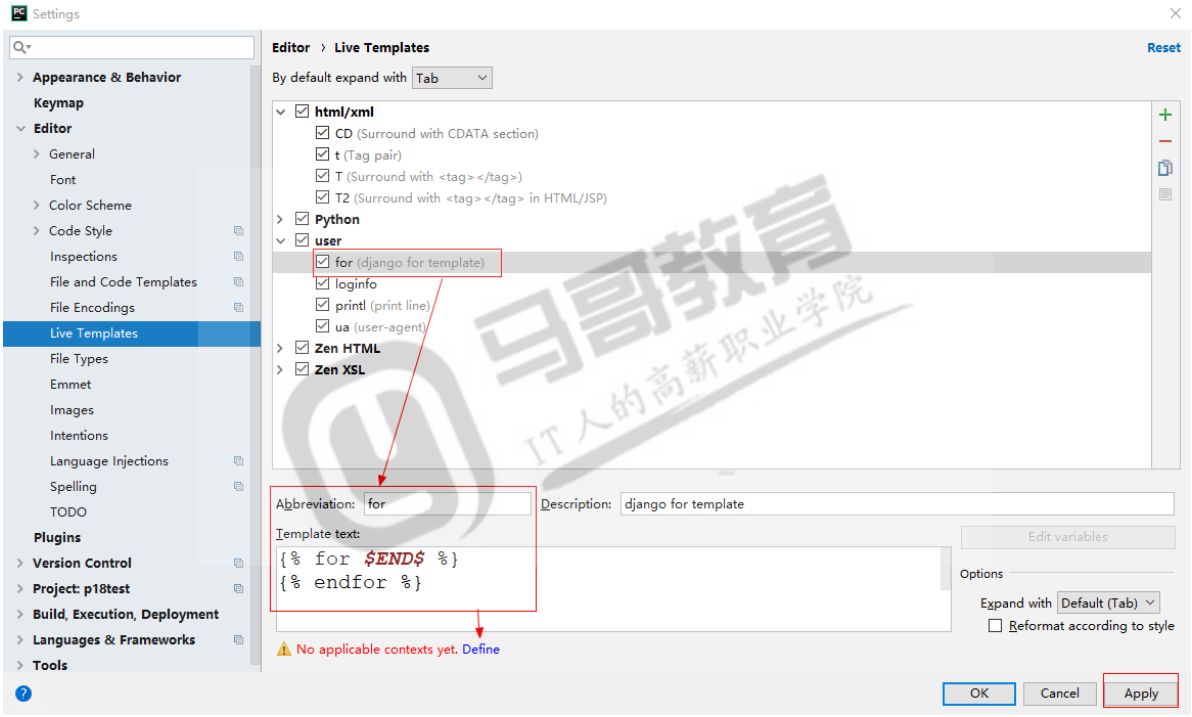
```
{% for athlete in athlete_list reversed %}
...
{% empty %}
... 如果被迭代的列表是空的或者不存在，执行empty
{% endfor %}
```

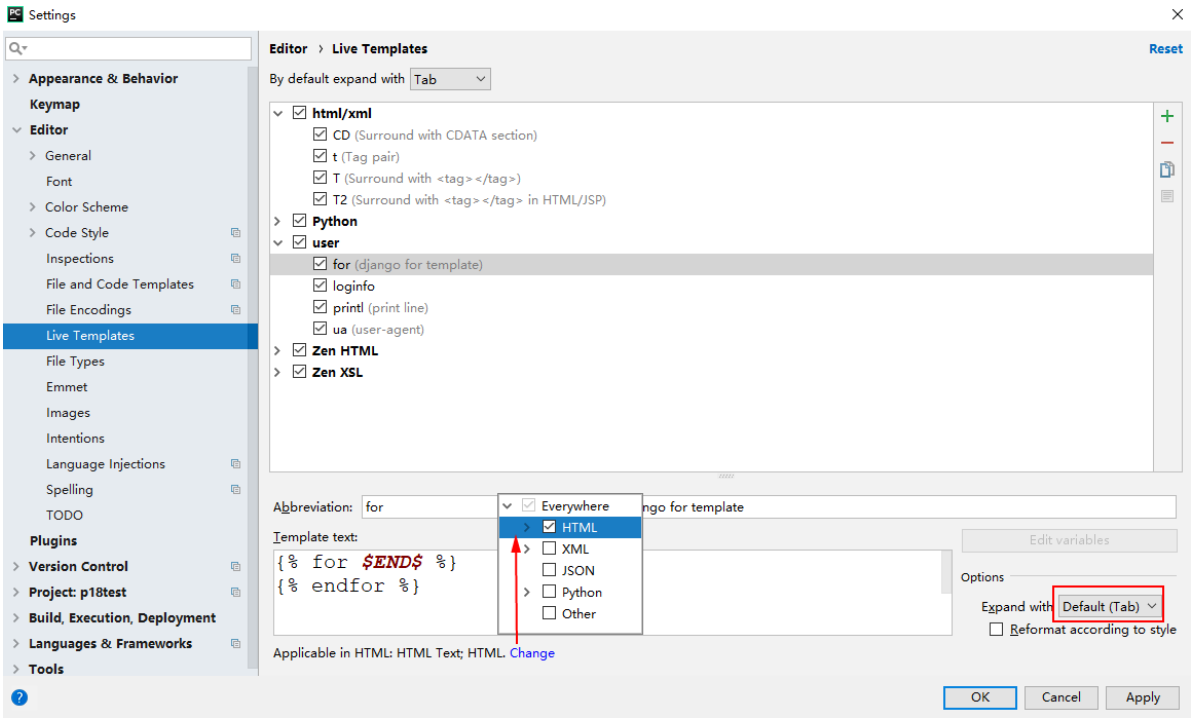
可以嵌套使用 {% for %} 标签：

```
{% for athlete in athlete_list %}
    <h1>{{ athlete.name }}</h1>
    <ul>
    {% for sport in athlete.sports_played %}
        <li>{{ sport }}</li>
    {% endfor %}
    </ul>
{% endfor %}
```

Pycharm模板定义


Settings
Editor › Live Templates
Reset
By default expand with Tab
Appearance & Behavior
Keymap
Editor
General
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Emmet
Images
Intentions
Language Injections
Spelling
TODO
Plugins
Version Control
Project: p18test
Build, Execution, Deployment
Languages & Frameworks
Tools
html/xml
CD (Surround with CDATA section)
t (Tag pair)
T (Surround with <tag></tag>)
T2 (Surround with <tag></tag> in HTML/JSP)
Python
user
loginfo
printl (print line)
ua (user-agent)
Zen HTML
Zen XSL
1. Live Template
2. Template Group...
Abbreviation: loginfo
Description:
Template text:
import logging
FORMAT = "%(asctime)s %(threadName)s %(thread)d %(messa
Edit variables
Options
Expand with Default (Tab)
Reformat according to style
Applicable in Python; Python: class. Change
OK
Cancel
Apply


Settings
Editor › Live Templates
Reset
By default expand with Tab
Appearance & Behavior
Keymap
Editor
General
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Emmet
Images
Intentions
Language Injections
Spelling
TODO
Plugins
Version Control
Project: p18test
Build, Execution, Deployment
Languages & Frameworks
Tools
html/xml
CD (Surround with CDATA section)
t (Tag pair)
T (Surround with <tag></tag>)
T2 (Surround with <tag></tag> in HTML/JSP)
Python
user
for (django for template)
loginfo
printl (print line)
ua (user-agent)
Zen HTML
Zen XSL
Abbreviation: for
Description: django for template
Template text:
{% for END %}
{% endfor %}
Edit variables
Options
Expand with Default (Tab)
Reformat according to style
No applicable contexts yet. Define
OK
Cancel
Apply



按照上述方法可以定义诸多Django模板标签。

testfor.html模板

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>测试for</title>
</head>
<body>
字典是dict(zip('abced', range(1,6)))
<ul>
{% for k,v in dct.items %}
    <li>{{forloop.counter}} {{k}} {{v}}</li>
{% endfor %}
</ul>
<hr>

<ul>
{% for k,v in dct.items %}
    <li>{{forloop.counter0}} {{k}} {{v}}</li>
{% endfor %}
</ul>
<hr>

<ul>
{% for k,v in dct.items %}
    {{ forloop.first }}
    {{ forloop.last }}
    <li>{{forloop.revcounter0}} {{k}} {{v}}</li>
{% endfor %}
</ul>
<hr>

```

```
<ul>
{% for k,v in dct.items %}
    <li>{{forloop.revcounter}} {{k}} {{v}}</li>
{% endfor %}
</ul>
<hr>

</body>
</html>
```

ifequal/ifnotequal 标签
{% ifequal %} 标签比较两个值，当他们相等时，显示在 {% ifequal %} 和 {% endifequal %} 之中所有的值。
下面的例子比较两个模板变量 user 和 currentuser :

```
{% ifequal user currentuser %}
    <h1>Welcome!</h1>
{% endifequal %}
```

和 {% if %} 类似， {% ifequal %} 支持可选的 {% else%} 标签:

```
{% ifequal section 'sitenews' %}
    <h1>Site News</h1>
{% else %}
    <h1>No News Here</h1>
{% endifequal %}
```

其他标签

csrf_token 用于跨站请求伪造保护，防止跨站攻击的。
{% csrf_token %}

3 注释标签

单行注释 {# #}。
多行注释 {% comment %} ... {% endcomment %}.

```
{# 这是一个注释 #}

{% comment %}
这是多行注释
{% endcomment %}.
```

4 过滤器

模板过滤器可以在变量被显示前修改它。

语法
`{{ 变量|过滤器 }}`
过滤器使用管道字符 | ，例如 `{{ name|lower }}` ， `{{ name }}` 变量被过滤器 lower 处理后，文档大写转换文本为小写。

过滤管道可以被**套接**，一个过滤器管道的输出又可以作为下一个管道的输入。
例如 `{{ my_list|first|upper }}` ，将列表第一个元素并将其转化为大写。

过滤器传参

有些过滤器可以传递参数，过滤器的参数跟随冒号之后并且总是以双引号包含。

例如：`{{ bio|truncatewords:"30" }}`，截取显示变量 bio 的前30个词。

`{{ my_list|join:"," }}`，将my_list的所有元素使用,逗号连接起来

其他过滤器

过滤器	说明	举例
cut	切掉字符串中的指定字符	{{ value | cut:"b" }}
lower	转换为小写	
upper	转换为大写	
truncatewords	保留指定个数的**单词**	{{ bio | truncatewords:"30" }}
join	对序列拼接	{{ d.e | join:"//" }}
first	取序列第一个元素	
last	取序列最后元素	
yesno	变量可以是True、False、None yesno的参数给定逗号分隔的三个值，返回3个值中的一个。 True对应第一个 False对应第二个 None对应第三个 如果参数只有2个，None等效False处理	{{ value | yesno:"yeah,no,maybe" }}
add	加法。参数是负数就是减法	数字加{{ value | add:"100"}} 列表合并{{mylist | add:newlist}}
divisibleby	能否被整除	{{ value | divisibleby:"3" }}能被3整除返回True
addslashes	在反斜杠、单引号或者双引号前面加上反斜杠	{{ value | addslashes }}
length	返回变量的长度	{% if my_list | length > 1 %}
default	变量等价False则使用缺省值	{{ value | default:"nothing" }}
default_if_none	变量为None使用缺省值	{{ value | default_if_none:"nothing" }}
date	格式化 date 或者 datetime 对象	Y 2000 年 m 01~12 月 d 01~31 日 y 20年 n 1~12 月 j 1~31 日

```
{{ mydict.d.upper }} 调用字符串对象upper函数
{{ mydict.d|upper }} 字符串通过过滤器upper转成大写
{{ mydict.d.0 }} {{ mydict.d|first }} 0表示[0]，first是过滤器
{{ mydict.d|join:',' }}
{{ mydict.b|yesno:'nonzero,zero' }}
{{ mydict.a|add:'11' }}
{{ mydict.a|add:mydict.a }}
{{ mydict.a|add:'11'|divisibleby:3 }}
{{ mydict.a|divisibleby:3 }}
```

```
{{ mydict.c|length }}

{% if mydict.c|length > 50 %}
{{ mydict.c }}
{% endif %}

{{ mydict.date|date:'Y-n-j Y-m-d' }}
```

时间的格式字符查看https://docs.djangoproject.com/en/2.2/ref/templates/builtins/#date

过滤器参考 https://docs.djangoproject.com/en/2.2/ref/templates/builtins/#

模板习题

1、奇偶行列表输出

使用下面字典my_dict的c的列表，在模板网页中列表ul输出多行数据

- 要求奇偶行颜色不同
- 每行有行号（从1开始）
- 列表中所有数据都增大100

```
from django.http import HttpResponse, HttpRequest
from django.shortcuts import render

def index(request:HttpRequest):
    """视图函数：请求进来返回响应"""
    my_dict = {
        'a':100,
        'b':0,
        'c':list(range(10,20)),
        'd':'abc',
        'date':datetime.datetime.now()
    }
    context = {'content':'www.magedu.com', 'my_dict':my_dict}
    return render(request, 'index.html', context)
```

模板中如何实现?

2、打印九九方阵

```
1*1=1 1*2=2 ... 1*9=9
...
9*1=1 9*2=18... 9*9=81
```

使用把上面所有的数学表达式放到HTML表格对应的格子中。如果可以，请实现奇偶行颜色不同。

静态

使用模板不免用到静态文件，如何引入静态文件呢?

首先看看静态配置是什么？提供一个视图函数，如下

```python
from django.shortcuts import render
from django.conf import settings

def test_index(request):
    print('~' * 30)
    static_info = list(filter(lambda x: x.find('STATIC')!=-1,
dir(settings)))
    for x in static_info:
        print(x, getattr(settings, x))
    print('~' * 30)
    return render(request, 'index.html', {'users':[1, 2, 'a', 'b']})
```

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
STATICFILES_DIRS []
STATICFILES_FINDERS ['django.contrib.staticfiles.finders.FileSystemFinder',
'django.contrib.staticfiles.finders.AppDirectoriesFinder']
STATICFILES_STORAGE django.contrib.staticfiles.storage.StaticFilesStorage
STATIC_ROOT None
STATIC_URL /static/
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

官方参考 https://docs.djangoproject.com/en/3.2/ref/settings/#settings-staticfiles

Django的静态需要在settings.py中配置

- settings中INSTALLED_APPS确保有django.contrib.staticfiles
- STATIC_URL = '/static/'，URL访问的逻辑路径前缀
- STATICFILES_DIRS = []，静态文件搜索路径
- STATICFILES_FINDERS，搜索器
    - django.contrib.staticfiles.finders.FileSystemFinder搜索STATICFILES_DIRS的目录
    - django.contrib.staticfiles.finders.AppDirectoriesFinder搜索每一个app下面的static子目录

```python
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    BASE_DIR / 'static'
]
```

模板中使用如下

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>马哥教育Django首页</title>
  {% load static %}
<!--  <script src="/static/js/jquery-3.6.0.min.js"></script>-->
  <script src="{% static 'js/jquery-3.6.0.min.js' %}"></script>
</head>
<body>
<div id="app"></div>
</body>
</html>
```

https://docs.djangoproject.com/en/3.2/ref/templates/builtins/#static

js/jquery-3.6.0.min.js可以

1. BASE_DIR 下static下的js/jquery-3.6.0.min.js
2. BASE_DIR 下employee下static下的js/jquery-3.6.0.min.js

它们逻辑路径都是/static/js/jquery-3.6.0.min.js，搜索到匹配文件就立即返回，不在继续寻找。