Django模板标签完整攻略

前面我们用了三节的内容重点讲解了模板语言中使用频率较高到的三个标签 if、for 以及 url。在本节我们将其余标签选择性的讲解，学习 Django 框架的时候，要学会抓住知识重点，这是提高学习效率的一种有效途径。

1. 模板标签攻略

1) 判断变量值是否相等

判断两个变量的值是否相等，也是一种很常见的需求，我们使用 ifequal 标签来进行判断，它 endifequal 标签成对出现，当然也可以使用 if 标签配合比较运算符来实现，所以这个标签也显的很鸡肋，Django官方的解释是，为了简化操作过程所以提供了这个标签，所以这就要看大家的喜好，自己决定是否使用它吧。它的语法格式如下：

{% ifequal n1 n2 %}
...代码块1
{% else %}
...代码块2
{% endifequal %}

n1 与 n2 的值相等的，那么就执行代码块 1，如果不等就执行代码块 2，示例如下：

```
In [1]: from django.template import Template,Context
In [2]: t=Template("""
   ...: {% ifequal n1 n2 %}
   ...:  <p>{{ n1 }} equal {{ n2 }}</p>
   ...: {% else %}
   ...:   <p>{{ n1 }} not equal {{ n2 }} </p>
   ...: {% endifequal %}
   ...: """)
   ...: t.render(Context({'n1':'python','n2':'python'}))
Out[3]: '<p>python equal python</p>'
```

此处的 n1与 n2 的值属于模板变量，如使用硬编码时也可以是整数、小数或者字符串，如下所示

{% ifequal variable 1 %}
{% ifequal variable 1.23 %}
{% ifequal variable "bar" %}
但是 Python 的字典类型、列表类型、布尔类型，不能用在 {% ifequal %} 中，以下是错误的使用方法：

{% ifequal variable True %}
{% ifequal variable [1, 2, 3] %}
{% ifequal variable {'key': 'value'} %}

2) extends模板继承标签
extends 标签表示模板继承的标签，它通常会与{ block }标签一起使用，有具体两种使用法:

- {% extends "base.html" %} 继承名为 "base.html" 的父模板；
- {% extends variable %} 使用 variable 变量表示的模版。

模板继承在 Django 模板语言中也是非常重要的知识，我们将会对其做单独的讲解。

3) load加载标签或过滤器
load 标签用于在模板中加载自定义模板标签集或者过滤器，可以加载项目中的静态文件，格式使用如下：

{% load static %} 加载静态文件

4) ifchanged标签的使用
从循环的最后一次迭代检查值是否已更改。{% ifchanged %} 标签每次循环都将对应的值与上一次值进行对比，若发现前后两者的值不同，就会将更改后的值显示出来，我们通过具体的例子进行说明：

```
In [1]: t=Template("""
   ...: {% for name in webnames %}
   ...:     {% ifchanged %}
   ...:     {{name.1|add:'ioe'}}
   ...:     {% endifchanged %}
   ...: {% endfor %}
```

```
    ...: """)
    ...: t.render(Context({'webnames':[['Python','Flask'],'java','c语言']}))
Out[1]: '¥n¥n Flaskioe¥n aioe¥n 语ioe ¥n¥n'
```

通过输出结果可以看出， ifchanged 标签将更改后的值进行了显示，而其余未更改的部分则没有进行输出。add 为模板的过滤器，这里表示给 name 的相对应变量值进行了字符串拼接的操作。

5) csrf_token标签使用

一个防止 CSRF 攻击（跨站点请求伪造）的标签，对于 CSRF 攻击可以参见百度百科《跨站请求伪造》自行了解，不过此标签在后续相关章节还会使用到。

6) cycle标签的使用

每次遇到此标签时，都会产生一个 cycle 自带的参数。第一个参数在第一次遇到时产生，第二个参数在第二次遇到时产生，依此类推。一旦所有参数用尽，标记将循环到第一个参数并再次产生它。cycle 标签在循环中特别有用：

```
#cycle的参数为字符串，用引号引起来
In [17]: t=Template("""
    ...: {% for i in some_list %}
    ...:     <tr class="{% cycle 'row1' 'row2' %}">
    ...:         <p>{{ i }}</p>
    ...:     </tr>^M
    ...: {% endfor %}
    ...: """)
    ...: t.render(Context({'some_list':['Python','Flask']}))
Out[17]: '¥n¥n<tr class="row1">¥n<p>Python</p></tr> <tr class="row2">¥n<p>Flask</p></tr>¥n¥n'
#cycle参数为变量，需要用字典为其赋值
In [18]: t=Template("""
    ...: {% for i in some_list %}
    ...:     <tr class="{% cycle rowvalue1 rowvalue2 %}">
    ...:         <p>{{ i }}</p>
    ...:     </tr>
    ...: {% endfor %}
    ...: """)
    ...: t.render(Context({'some_list':['Python','Flask'],'rowvalue1':'row1','rowvalue2':'row2'}))
Out[18]: '¥n¥n<tr class="row1">¥n<p>Python</p>¥n</tr>  <tr class="row2">¥n<p>Flask</p>¥n</tr>¥n¥n'
```

字符串和变量也可以在 cycle 标签中混合使用，我们就不加赘述。在某些情况下，你可能只希望引用循环的当前值而不包含下一个值。为此，只需使用“ as”为标签起个别名即可，格式如下所示：

{% cycle 'row1' 'row2' as rowcolors %}

当需要在模板中插入被引用的当前值的时候，我们可以通过别名来引用，将其作为上下文的变量。如下所示：

```
<tr>
    <td class="{% cycle 'row1' 'row2' as rowcolors %}">...</td>
```

```
    <td class="{{ rowcolors }}">...</td>
</tr>
<tr>
    <td class="{% cycle rowcolors %}">...</td>
    <td class="{{ rowcolors }}">...</td>
</tr>
```

cycle 标签也可与 resetcycle 标签配合使用，resetcycle 表示重置上一个循环，遇到此标签时，将从 cycle 标签的第一个参数重新开始， cycle 还提供了关键字 silent，正如它英语含义一样，它表示不输入参数值，只显示循环周期的内容，即不会给 class 属性赋值。实例如下所示：

```
In [20]: t=Template("""
    ...: {% for o in some_list %}
    ...:     <tr class="{% cycle rowvalue1 rowvalue2 as rowcolor silent %}">
    ...:         <p>{{ o }}</p>
    ...:     </tr>
    ...: {% endfor %}
    ...: """)
    ...: t.render(Context({'some_list':['Python','Flask'],'rowvalue1':'row1','rowvalue':'row2'},autoescape=False ))
Out[20]: '¥n¥n    <tr class="">¥n<p>Python</p>¥n</tr>  <tr class="">¥n  <p>Flask</p>¥n </tr>¥n¥n'
```

2. 模板语言的注释

如同 HTML 和其他的语言例如 Python 一样，Django 模板系统也允许注释。 注释使用{# #} 的格式，示例如下：

{# This is a comment #}

注释的内容不会在模板渲染时输出，但是在很多情况下，由于要注释的内容非常多并且需要跨行，比如要增加代码逻辑描述等，这时使用这方式就不是很方便。为了解决上述问题，Django 提供了 comment 标签，用来快速实现多行注释，使用方法如下：

{% comment %}
...要被注释的内容放在两个标签中间
{% endcomment %}

3. 国际化标签

Django 还提供了国际化模板标签，以控制国际化的各个方面 。这些标签允许对翻译、格式和时区转换进行精细控制。如下所示：

1) i18n语言转换

该库允许在模板中指定可翻译文本。要启用它，请将设置 USE_I18N=True，然后加载使用 {% load i18n %}

2) l10n本地化

该库可控制模板中值的本地化。您只需要使用即可加载库，但通常会设置为 USE_L10N=True，以便默认情况下本地化处于活动状态。使用 {% load l10n %} 加载。

3) tz 时区转换

该库可控制模板中的时区转换。像一样l10n，但通常还需要将其设置为 USE_TZ=True 默认情况下转换为本地时间。使用 {% load tz %} 加载。

提示：使用国际化标签，需要更改 setting.py 配置文件，根据上述要求进行相应更改即可。
Django 官网提供了所有内置标签以及第三方标签库的使用方法，如果大家感兴趣可以参见如下网址：
https://docs.djangoproject.com/en/2.2/ref/templates/builtins/，虽然有些标签不经常使用，但是可以作为知识拓展进行了解。