



Network Programming

- 網路程式設計 -

Django視圖與模版

授課教師：張珀銀

## 03

## 視圖與模版

1. Django 的 Framework 架構
2. 視圖與模版
3. Template 語言
4. 課堂練習

# 1 Django 的 Framework 架構

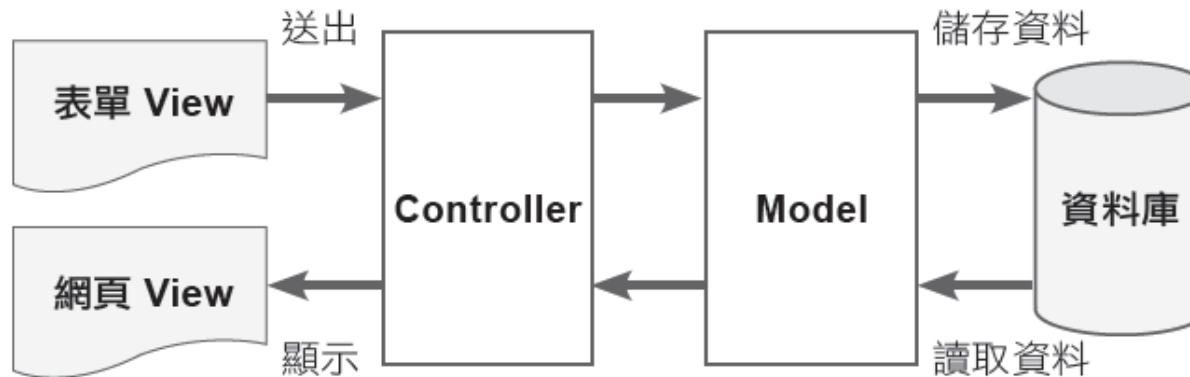
## 1.1 認識MVC

MVC 是軟體設計的一種架構，它將軟體系統切開分為三個部分，分別是Model、View 及Controller。

Model 模型：代表商業邏輯與資料庫存取有關的程式。

View 視圖：代表輸入和輸出畫面的呈現。

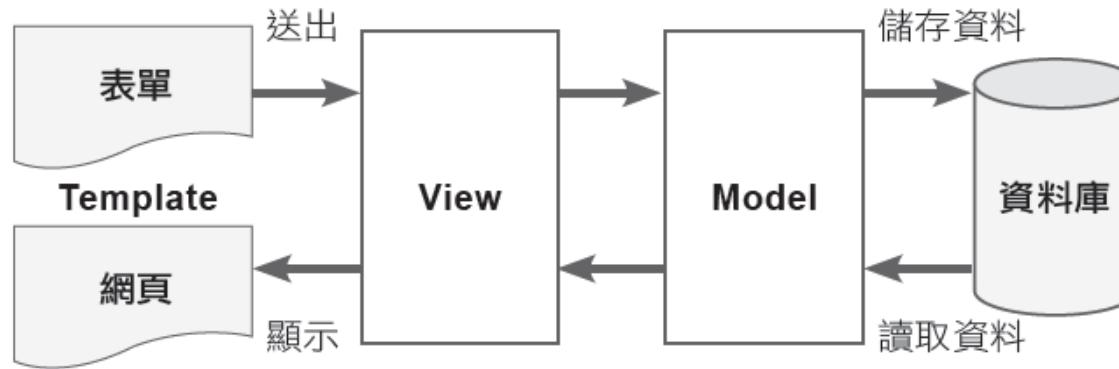
Controller 控制：介於Model 及View 之間，判斷該呼叫哪一個Model 存取資料庫並透過哪一個View 介面輸入表單或顯示資料。



# 1.2 Django 的 MTV 架構

Django 架構也是採用模型(Model)、視圖(View) 和控制(Controller) 分開的架構，但Django 中稱為MTV 架構，和MVC 稍有差異。

MTV 架構的M 是 Model、T 是Template，而V 則是View。



為了避免混淆，以下面表格來對照。

任務	MVC 架構	MTV 架構
資料庫存取	M=Model= 模型	M=Model= 模型
輸入表單或顯示資料	V=View= 視圖	T=Template= 模版
控制與整合	C=Controller= 控制	V=View= 視圖

## 3.2 視圖與模版

### 3.2.1 建立視圖與模版

#### 定義url

firstproject\urls.py

```
from myapp.views import hello4
urlpatterns = [
    .....略
    path('hello4/<str:username>', hello4),
]
```

#### 定義View 視圖

myapp\views.py

```
def hello4(request,username):
    now=datetime.now()
    return render(request,"hello4.html",locals())
```

# 建立Template 模版

## 建立Template 模版

**templates\hello4.html**

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>顯示圖片的模版 </title>
    {% load staticfiles %}
    .....略
</head>
<body>
    <div id="home">
        
        <span class="info"> 歡迎光臨： {{username}}</span>
        <h2> 現在時刻： {{now}} </h2>
    </div>
</body>
</html>
```

## 3.2.2 傳遞變數到Template 模板檔案

### 傳遞字典到顯示模版

render 的語法如下：

```
render(request, template_name, context)
```

第一個參數是HttpRequest 物件，第二個參數是模板名稱，最後一個參數是字典。例如：傳遞字典「{"no":1}」給顯示模版<dice.html>。

```
return render(request,"dice.html", {"no":1})
```

也可以傳遞多筆不同型別的字典，例如：傳遞字典 {"name":"Amy","age":20}。

```
return render(request,"dice.html", {"name":"Amy", "age":20})
```

使用字典變數也沒問題，例如：傳遞 dict1 字典變數。請注意：這種方式在模版中取得其內容的方式是 {{ 鍵}}，如{{name}}，而不是 {{dict1.name}}。

```
dict1={"name":"Amy", "age":20}
return render(request,"dice.html",dict1)
```

請將本章範例passdata 專案複製到<C:\example> 資料夾中，開啟命令提示字元視窗切換到passdata 專案資料夾。

```
cd C:\example\passdata
```

以<manage.py> 即可啟動伺服器。

```
python manage.py runserver
```

在 urlpatterns 中定義 「/dice/」、「/dice2/」、「/dice3/」、「/show/」、「/filter/」 分別執行 dice、dice2、dice3、show 和filter 自訂函式。

### passdata\urls.py

```
from passdataapp.views import dice,dice2,dice3,show,filter
urlpatterns = [
    path('admin/', admin.site.urls),
    path('dice/', dice),
    path('dice2/', dice2),
    path('dice3/', dice3),
    path('show/', show),
    path('filter/', filter),
]
```

## 範例：傳遞變數到 Template 模板

請執行「<http://127.0.0.1:8000/dice/>」，將會以亂數 1~6 顯示骰子點數。



### passdataapp\views.py

```
from django.shortcuts import render
import random # 加入 random 模組
def dice(request):
    no=random.randint(1,6) # 1~6
    return render(request,"dice.html",{"no":no})
```

在<dice.html> 模版中以{{no}} 顯示字典。

### templates\dice.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>擲骰子 </title>
</head>
<body>
    <h1> 點數 : {{no}} </h1>
</body>
</html>
```

## 使用locals() 函式

如果變數太多，也可以在render 函式中的第 3 個參數使用locals() 函式，locals() 函式可以將所有區域變數轉換成字典，這樣就能一次傳遞所有變數。不過請注意：locals() 函式只能傳遞區域變數，如果是global 變數將不會被處理。

```
no=1  
dict1={"name":"Amy","age":20}  
return render(request,"dice.html",locals())
```

## 模版顯示變數

在模版中可以顯示接收的變數，必須以「變數」為鍵，語法如下：

```
{{ 變數 }}
```

例如：顯示變數no。(註：「no=1」已由locals() 轉換為「{"no":1}」)

```
{{no}}
```

如果是字典變數，前面必須加上字典變數名稱，語法如下：

```
 {{ 字典變數 . 鍵 }} 
```

例如：顯示dict1 字典name 鍵的鍵值內容，由於

「dict1={"name":"Amy","age":20}」已由locals() 轉換為  
「"dict1":{"name":"Amy","age":20}」字典，因此可以下列語法取得 name 鍵的  
鍵值內容。

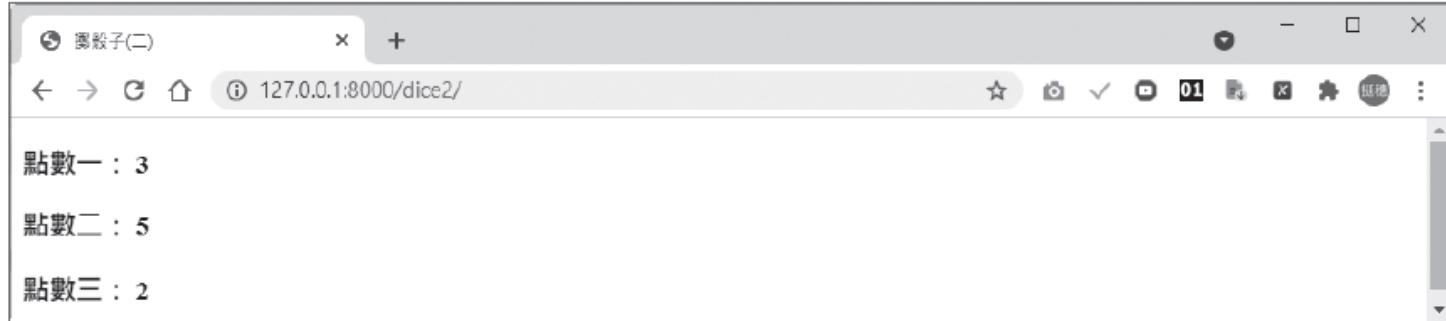
```
 {{ dict1.name }} 
```

如果是串列變數，則以「串列變數. 索引」的格式取得。例如：顯示list1 串列的  
第 1個元素內容。

```
 {{ list1.0 }} 
```

## 範例：使用locals() 傳遞變數到Template 模板(一)

請執行「<http://127.0.0.1:8000/dice2/>」，將會以亂數1~6 分別顯示3 個骰子點數。



## 範例：使用locals() 傳遞變數到Template 模板(二)

請執行「<http://127.0.0.1:8000/dice3/>」，將會顯示玩家名稱、骰子點數和累計數。



### 3-3 Template 語言

Template 模版的組成：

名稱	說明	範例
變量	將 <b>views</b> 傳送內容顯示在模版指定的位置上。	<code>{{username}}</code>
標籤	<b>if</b> 條件指令和 <b>for</b> 迴圈指令。	<code>{% if found %}</code> <code>{ % for item in items %}</code>
單行註解	語法： <code>{# 註解文字 #}</code> 。	<code>{# 這是註解文字 #}</code>
多行註解	語法： <code>{% comment %}</code> 註解文字一 註解文字二 ... <code>{% endcomment %}</code>	<code>{{% comment %}}</code> 註解文字第一列 ... 註解文字第 <b>n</b> 列 <code>{% endcomment %}</code>
文字	HTML 標籤或文字。	<code>&lt;title&gt; 顯示的模版 &lt;/title&gt;</code>

### 3.3.1 變量

變量類型	Template 語法	Python 語法	說明
字典	<code>{{dict1.name}}</code>	<code>dict1[name]</code>	<code>name</code> 是字典的鍵。
屬性	<code>{{obj1.name}}</code>	<code>obj1.name</code>	<code>name</code> 是 <code>obj1</code> 物件的屬性。
方法	<code>{{obj1.show}}</code>	<code>obj1.show()</code>	<code>show()</code> 是 <code>obj1</code> 物件的方法。
串列	<code>{{list1.0}}</code>	<code>list1[0]</code>	<code>list1</code> 是串列。例如： <code>list1=["a","b","c"]</code> 。

## 3.3.2 標籤

### 條件指令

#### 單向判斷式

```
{% if 條件 %}  
    程式區塊  
{% endif %}
```

#### 雙向判斷式

```
{% if 條件 %}  
    程式區塊一  
{% else %}  
    程式區塊二  
{% endif %}
```

#### 多向判斷式

```
{% if 條件一 %}  
    程式區塊一  
{% elif 條件二 %}  
    程式區塊二  
{% elif 條件三 %}  
    程式區塊三  
...  
[{% else %}  
    程式區塊 else]  
{% endif %}
```

### for 迴圈指令

```
{% for 變數 in 串列 %}  
    程式區塊  
[{% empty %}]  
    程式區塊 empty  
{% endfor %}
```

例如：定義list1 串列為 list1 = range(1,6)。

### views.py

```
def show(request):
    list1 = range(1,6)
    return render(request,"show.html",locals())
```

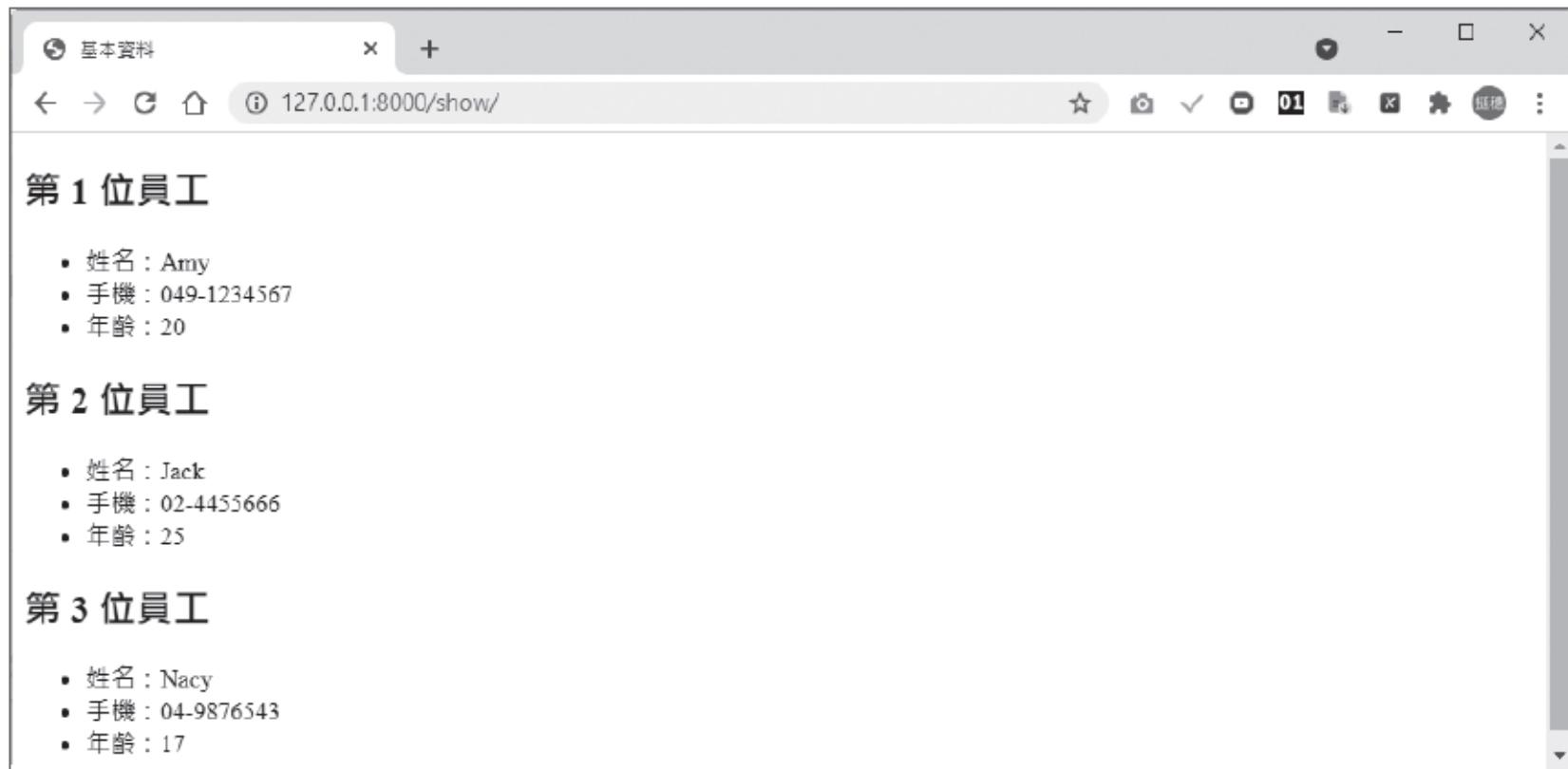
在樣版中以for 迴圈顯示list1，執行結果為「1,2,3,4,5,」。

### show.html

```
{% for i in list1 %}
    {{i}},
{% empty %}
    沒有任何資料
{% endfor %}
```

# 範例：模版中接收串列資料並依序顯示

請執行「<http://127.0.0.1:8000/show/>」，將會依序顯示串列中的資料。



# forloop 變量及其屬性

Template 的for 迴圈提供forloop 屬性，當作計數器。

forloop 屬性如下：

forloop 屬性	說明
forloop.counter	由 1 開始遞增到迭代總數。
forloop.counter0	由 0 開始遞增到迭代總數。
forloop.revcounter	由串列元素總數開始遞減到 1。
forloop.revcounter0	由串列元素總數開始遞減到 0。
forloop.first	判斷是否是第一次 for 迴圈，其值為 True 或 False。
forloop.last	判斷是否是最後一次 for 迴圈，其值為 True 或 False。
forloop.parentloop	父迴圈 ( 上一層迴圈 ) 的 foorloop 。

例如：延續上例，由1 開始顯示資料第幾位員工。

執行結果：

The screenshot shows a web browser window with the title '基本資料' and the URL '127.0.0.1:8000/show/'. The browser interface includes standard controls like back, forward, and search, along with a toolbar with various icons.

**第 1 位員工**

- 姓名 : Amy
- 手機 : 049-1234567
- 年齡 : 20

**第 2 位員工**

- 姓名 : Jack
- 手機 : 02-4455666
- 年齡 : 25

**第 3 位員工**

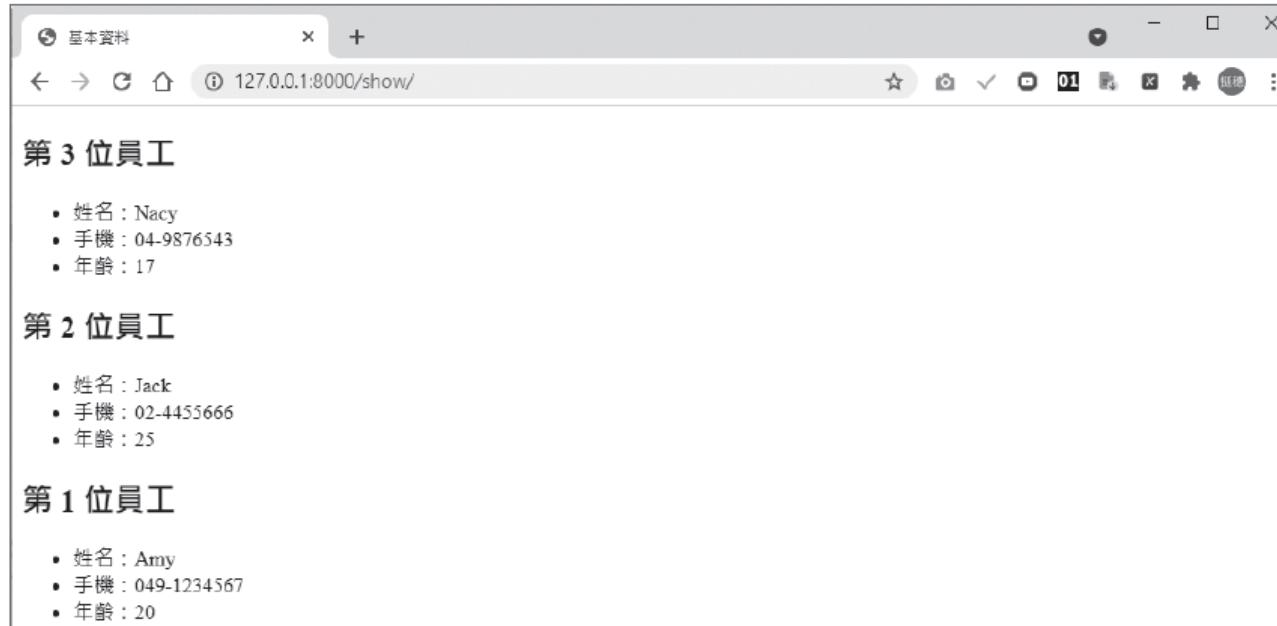
- 姓名 : Nacy
- 手機 : 04-9876543
- 年齡 : 17

# for 迴圈倒轉

要將串列資料倒轉輸出，通常有兩種解決的方法。

1. 在<views.py> 視圖函式將取得的資料倒轉。
2. 在Template 模版的for 指令中加上reversed，例如：「{% for person in personsreversed %}」將 persons 串列以倒轉方式迭代。

例如：將上例 <show.html> 的第9-10 列改為下列程式碼，資料和計數器都會反轉。



### 3.3.3 過濾器 Filter

過濾器(Filter)是指將變數值以Django函數處理後輸出，常用的過濾器如下表

過濾器名稱	範例	說明
add	<code>{{ value add:"2" }}</code>	如果 <code>value</code> 是數值，例如 <code>value=4</code> ，顯示 <code>4+2</code> 的結果「6」。若 <code>value="A"</code> 是字串，顯示 " <code>A</code> "+"2" 的結果「A2」。
addslashes	<code>{{ value addslashes }}</code>	如果 <code>value = "I'm using Django"</code> ，加上跳脫字元後顯示結果為「"I\'m using Django"」。
capfirst	<code>{{ value capfirst }}</code>	將 <code>value</code> 字串第 1 個字元轉換為大寫，若字元不是字母將不予處理。
cut	<code>{{ value cut:" 指定字串 " }}</code>	移除 <code>value</code> 字串所有指定的字串。例如： <code>{{ value cut:" " }}</code> 移除所有空白字元。
date	<code>{{ value date:"D d M Y" }}</code>	假如日期時間為「2017 年 6 月 13 日 15:10」，顯示結果為「星期二 13 六月 2017」。 <code>date</code> 過濾器格式太多，請自行參考官網說明。
default	<code>{{ value default:" 預設值 " }}</code>	如果 <code>value</code> 是空字串，將輸出預設值。

過濾器名稱	範例	說明
dictsort	<code>{{ value dictsort:"鍵" }}</code>	將字典依指定的「鍵」遞增排序。 例如： <code>{{ value dictsort:"name" }}</code> 依 name 遞增排序。
dictsortreversed	<code>{{ value dictsortreversed:"鍵" }}</code>	將字典依指定的「鍵」遞減排序。
divisibleby	<code>{{ value divisibleby:"3" }}</code>	若 value 可被 3 整除，傳回 True。
escape	<code>{{ value escape }}</code>	若 value 中含有 HTML 標籤， HTML 標籤將失去作用，被視為文 字輸出。 例如： <code>value=&lt;h1&gt;Hello&lt;/h1&gt;</code> 則 <code>&lt;h1&gt;</code> 標籤將失去作用，顯示結果 為「 <code>&lt;h1&gt;Hello&lt;/h1&gt;</code> 」。
filesizeformat	<code>{{ value filesizeformat }}</code>	顯示檔案大小，例如： <code>value =123456789</code> ，結果為 117.7 MB。
first	<code>{{ value first }}</code>	傳回 value 串列中第 1 個項目。 例如： <code>value= ['a12', 'b', 'c']</code> ， 傳回值為「 <code>a12</code> 」。
join	<code>{{ value join://" // " }}</code>	將串列元素以指定字元串接。 例如： <code>value=['a', 'b', 'c']</code> ， 傳回值為「 <code>"a // b // c"</code> 」。

過濾器名稱	範例	說明
last	<code>{{ value last }}</code>	傳回 <code>value</code> 串列中最後的項目。 例如： <code>value= ['a12', 'b', 'c']</code> ， 傳回值為「c」。
length	<code>{{ value length }}</code>	傳回 <code>value</code> 串列或字串的長度。 例如： <code>value= ['a12', 'b', 'c']</code> ， 傳回值為 3。 <code>value= "abcd"</code> ，傳回值為 4，若變數未定義則傳回 0。
length_is	<code>{{ value length_is:"4" }}</code>	若 <code>value</code> 長度為 4，傳回 True。
linebreaks	<code>{{ value linebreaks }}</code>	將文字內容的換行符號 ( <code>\n</code> ) 轉換為 HTML 的 <code>&lt;br /&gt;</code> 和 <code>&lt;p&gt;&lt;/p&gt;</code> 。 例如： <code>value="Joel\nis a slug"</code> ，傳 回值為「 <code>&lt;p&gt;Joel&lt;br /&gt;is a slug&lt;/p&gt;</code> 」。
linebreaksbr	<code>{{ value linebreaksbr }}</code>	將文字內容的換行符號 ( <code>\n</code> ) 轉換為 HTML 的 <code>&lt;br /&gt;</code> 。 例如： <code>value="Joel\nis a slug"</code> ，傳 回值為「 <code>Joel&lt;br /&gt;is a slug</code> 」。

過濾器名稱	範例	說明
linenumbers	<code>{{ value linenumbers }}</code>	將顯示文字前面加上行號。 例如： <code>value="Hello"</code> ，傳回值為「1.Hello」。
lower	<code>{{ value lower }}</code>	將字串轉換小寫。
make_list	<code>{{ value make_list }}</code>	將字串轉換為串列。 例如： <code>value = "123"</code> ，傳回值為 [1, 2, 3]。
random	<code>{{ value random }}</code>	以亂數方式取得串列的元素。
safe	<code>{{ value safe }}</code>	以 HTML 格式讀取字串。 例如： <code>value = "&lt;h1&gt;Hello&lt;/h1&gt;"</code> ，傳回值為 <h1> 網頁格式的文字「Hello」。
slice	<code>{{ lista slice:"2" }}</code>	傳回部份字串，例如： <code>lista= ['a', 'b', 'c']</code> ，傳回值為 [a, b]。
slugify	<code>{{ value slugify }}</code>	將字串中的空白符以 - 號取代。例如： <code>value="Good Bye"</code> ，傳回值為「Good-Bye」。

過濾器名稱	範例	說明
stringformat	<code>{{ value stringformat:"E" }}</code>	以科學記號表示。例如： <code>value=4</code> ，傳回值為「 <code>4.000000E+00</code> 」。
striptags	<code>{{ value striptags }}</code>	移除所有的 HTML 標籤。
title	<code>{{ value title }}</code>	將字串中每個 Word 中的第 1 個字元轉換為大寫。 例如： <code>value="my FIRST post"</code> ，傳回值為「 <code>My First Post</code> 」。
truncatechars	<code>{{ value truncatechars:9 }}</code>	將字串中多出的字串以「...」顯示。 例如： <code>value="Joel is a slug"</code> ，取前面 6 個字元再加「...」共 9 個字元， 傳回值為 " <code>Joel i...</code> "。
upper	<code>{{ value upper }}</code>	將字串轉換大寫。
wordcount	<code>{{ value wordcount }}</code>	傳回字串的 Word 數目。 例如： <code>value="Joel is a slug"</code> ，傳回值為 4。
yesno	<code>{{ value yesno: "是 , 否 , 取消 " }}</code>	依 <code>value</code> 內容是 <code>True</code> 、 <code>False</code> 或 <code>None</code> 分別顯示「是」、「否」或「取消」。

# 範例：使用 filter 過濾器

請執行「<http://127.0.0.1:8000/filter/>」，顯示過濾器測試的結果。



The screenshot shows a web browser window titled "filter". The address bar displays the URL "127.0.0.1:8000/filter/". The main content area contains the following text output from a Jinja2 template:

```
value=4,value|add:"2"= 6
value|stringformat:"E"=4.000000E+00
list1=[1, 2, 3],list1|length= 3
密碼正確
value2=False ,value2|yesno:"是,否,取消"-否
html=<h1>Hello</h1> ,html|safe=
```

Below this, the word "Hello" is displayed in large, bold black font.



**E**

Exercise 1: for 迴圈

# url.py

```
21     urlpatterns = [
22         path('admin/', admin.site.urls),
23         # path('hello4/<str:num1>', dice),
24         path('hello4/<str:username>', hello4),
25         path('dice/', dice),
26         path('hello3/', hello3),|
27         path('', index),
28         path('show/', show),|
29         path('filter/', filter),
30     ]
```

# views.py

```
def show(request):
    list1 = range(0, 6)
    return render(request, 'show.html', locals())
```

# show.py

```
1  !DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Show</title>
6  </head>
7  <body>
8  {% for i in list1 %}
9      {{ i }} <br>
10     {% empty %}
11         no data or missing variables name
12     {% endfor %}
13
14     </body>
15 </html>
```

# 結果

0  
1  
2  
3  
4  
5



## Exercise 2: forloop屬性

# url.py

```
19     from passdata.views import dice, hello4, index, hello3, filter, show, show2
20
21     urlpatterns = [
22         path('admin/', admin.site.urls),
23         # path('hello4/<str:num1>', dice),
24         path('hello4/<str:username>', hello4),
25         path('dice/', dice),
26         path('hello3/', hello3),
27         path('', index),
28         path('show/', show),
29         path('show2/', show2), # Line 29 highlighted with a red box
30         path('filter/', filter),
31     ]
```

# views.py

```
33     def show2(request):
34         person1 = {'name': 'Jack', 'phone': '0932', 'age': 21}
35         person2 = {'name': 'Mary', 'phone': '0952', 'age': 23}
36         person3 = {'name': 'Tom', 'phone': '0937', 'age': 22}
37         persions = [person1, person2, person3]
38         return render(request, 'show2.html', locals())
```

# show2.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Show</title>
6 </head>
7 <body>
8     {% for i in persions %}
9         <h2>第{{ forloop.counter }}位員工</h2>
10        <ui>
11            <li>姓名: {{ i.name }}</li>
12            <li>電話: {{ i.phone }}</li>
13            <li>年齡: {{ i.age }}</li>
14        </ui>
15        {% empty %}
16            no data or missing variables name
17    {% endfor %}
18
19    </body>
20 </html>
```

# Results

## 第1位員工

- 姓名: Jack
- 電話: 0932
- 年齡: 21

## 第2位員工

- 姓名: Mary
- 電話: 0952
- 年齡: 23

## 第3位員工

- 姓名: Tom
- 電話: 0937
- 年齡: 22



## Exercise 3: forloop 倒轉

# show3.html

```
1  




2      <html lang="en">
3          <head>
4              <meta charset="UTF-8">
5              <title>Show</title>
6          </head>
7          <body>
8              {% for i in persions reversed %}
9                  <h2>第{{ forloop.counter }}位員工</h2>
10                 <ul>
11                     <li>姓名: {{ i.name }}</li>
12                     <li>電話: {{ i.phone }}</li>
13                     <li>年齡: {{ i.age }}</li>
14                 </ul>
15             {% empty %}
16             no data or missing variables name
17         {% endfor %}
18
19     </body>
20 </html>
```

## 第1位員工

- 姓名: Tom
- 電話: 0937
- 年齡: 22

## 第2位員工

- 姓名: Mary
- 電話: 0952
- 年齡: 23

## 第3位員工

- 姓名: Jack
- 電話: 0932
- 年齡: 21



**E**

## Exercise 4: Filter

# url.py

```
16     from django.contrib import admin
17     from django.urls import path
18
19     from passdata.views import dice, hello4, index, hello3, filter
20
21     urlpatterns = [
22         path('admin/', admin.site.urls),
23         # path('hello4/<str:num1>', dice),
24         path('hello4/<str:username>', hello4),
25         path('dice/', dice),
26         path('hello3/', hello3),
27         path('/', index),
28         path('filter/', filter),
29     ]
```

# views.py

```
28     def filter(request):
29         value = 4
30         list1 = [1, 2, 3]
31         pw = "芝麻開門"
32
33         html = "<h1>Hello</h1>"
34         value2 = False
35         return render(request, "filter.html", locals())
```

# Filter.html

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset='utf-8'>
5     <title>filter</title>
6 </head>
7 <body>
8     <h3>
9         value={{ value }},value|add:"2"= {{ value|add:"2" }} <br />
10        value|stringformat:"E"={{ value|stringformat:"E" }} <br />
11        list1={{ list1 }},list1|length= {{ list1|length }} <br />
12
13        {% if pw == "芝麻開門" %} ←— 字串比對 →
14            密碼正確 <br />
15        {% else %}
16            密碼錯誤 <br />
17        {% endif %}
18
19        value2={{ value2 }} ,value2|yesno:"是,否,取消"={{ value2|yesno:"是,否,取消" }} <br />
20        html={{ html }},html|safe= {{ html|safe }}
21        </h3>
22    </body>
23 </html>
```

# Result

```
value=4,value|add:"2"= 6  
value|stringformat:"E"=4.000000E+00  
list1=[1, 2, 3],list1|length= 3  
密碼正確  
value2=False ,value2|yesno:"是,否,取消"=否  
html=<h1>Hello</h1> ,html|safe=
```

Hello