Django - Template System

Django makes it possible to separate python and HTML, the python goes in views and HTML goes in templates. To link the two, Django relies on the render function and the Django Template language.

The Render Function

This function takes three parameters −

Request − The initial request.
The path to the template − This is the path relative to the TEMPLATE_DIRS option in the project settings.py variables.
Dictionary of parameters − A dictionary that contains all variables needed in the template. This variable can be created or you can use locals() to pass all local variable declared in the view.

Django Template Language (DTL)

Django’s template engine offers a mini-language to define the user-facing layer of the application.

Displaying Variables

A variable looks like this: {{variable}}. The template replaces the variable by the variable sent by the view in the third parameter of the render function. Let's change our hello.html to display today’s date −

hello.html

```
<html>

   <body>
      Hello World!!!<p>Today is {{today}}</p>
   </body>

</html>
```

Then our view will change to −

```
def hello(request):
   today = datetime.datetime.now().date()
   return render(request, "hello.html", {"today" : today})
```

We will now get the following output after accessing the URL/myapp/hello −

```
Hello World!!!
Today is Sept. 11, 2015
```

As you have probably noticed, if the variable is not a string, Django will use the __str__ method to display it; and with the same principle you can access an object attribute just like you do it in Python. For example: if we wanted to display the date year, my variable would be: {{today.year}}.

Filters

They help you modify variables at display time. Filters structure looks like the following: {{var|filters}}.

Some examples −

{{string|truncatewords:80}} − This filter will truncate the string, so you will see only the first 80 words.
{{string|lower}} − Converts the string to lowercase.
{{string|escape|linebreaks}} − Escapes string contents, then converts line breaks to tags.

You can also set the default for a variable.

Tags

Tags lets you perform the following operations: if condition, for loop, template inheritance and more.

Tag if

Just like in Python you can use if, else and elif in your template −

```
<html>
   <body>

      Hello World!!!<p>Today is {{today}}</p>
      We are
      {% if today.day == 1 %}

      the first day of month.
      {% elif today.day == 30 %}

      the last day of month.
      {% else %}

      I don't know.
      {%endif%}

   </body>
</html>
```

In this new template, depending on the date of the day, the template will render a certain value.

Tag for

Just like 'if', we have the 'for' tag, that works exactly like in Python. Let's change our hello view to transmit a list to our template −

```
def hello(request):
   today = datetime.datetime.now().date()

   daysOfWeek = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
   return render(request, "hello.html", {"today" : today, "days_of_week" : daysOfWeek})
```

The template to display that list using {{ for }} −

```
<html>
   <body>

      Hello World!!!<p>Today is {{today}}</p>
      We are
      {% if today.day == 1 %}

      the first day of month.
      {% elif today.day == 30 %}

      the last day of month.
      {% else %}
```

```
      I don't know.
      {%endif%}

      <p>
         {% for day in days_of_week %}
         {{day}}
      </p>

      {% endfor %}

   </body>
</html>
```

And we should get something like −

```
Hello World!!!
Today is Sept. 11, 2015
We are I don't know.
Mon
Tue
Wed
Thu
Fri
Sat
Sun
```

Block and Extend Tags

A template system cannot be complete without template inheritance. Meaning when you are designing your templates, you should have a main template with holes that the child's template will fill according to his own need, like a page might need a special css for the selected tab.

Let’s change the hello.html template to inherit from a main_template.html.

main_template.html

```
<html>
   <head>

      <title>
         {% block title %}Page Title{% endblock %}
      </title>

   </head>

   <body>
```

```
      {% block content %}
         Body content
      {% endblock %}

   </body>
</html>
```

hello.html

```
{% extends "main_template.html" %}
{% block title %}My Hello Page{% endblock %}
{% block content %}

Hello World!!!<p>Today is {{today}}</p>
We are
{% if today.day == 1 %}

the first day of month.
{% elif today.day == 30 %}

the last day of month.
{% else %}

I don't know.
{%endif%}

<p>
   {% for day in days_of_week %}
   {{day}}
</p>

{% endfor %}
{% endblock %}
```

In the above example, on calling /myapp/hello we will still get the same result as before but now we rely on extends and block to refactor our code −

In the main_template.html we define blocks using the tag block. The title block will contain the page title and the content block will have the page main content. In home.html we use extends to inherit from the main_template.html then we fill the block define above (content and title).

Comment Tag

The comment tag helps to define comments into templates, not HTML comments, they won’t appear in HTML page. It can be useful for documentation or just commenting a line of code.

Revision #1
Created 14 December 2022 11:07:02 by Admin
Updated 14 December 2022 11:07:44 by Admin