Inside your Django app directory, create a module called templatetags and add an empty \_\_init\_\_.py file as shown in the below directory structure.

my\_app/

2├── \_\_init\_\_.py

3├── admin.py

4├── models.py

5├── templatetags/

6│ ├── \_\_init\_\_.py

7│ └── custom\_tags.py

8└── views.py

Next, open the custom\_tags.py file and add these two lines to get started with custom template tags and filters.

Next, open the custom\_tags.py file

from django import template

2

3register = template.Library()

Make custom\_tags available by loading it in templates.

1{% load custom\_tags %}

**Writing Custom Template Filters**

Filters are nothing but a Python function, which takes one or two arguments. For instance, in the filter {{var|filter\_f:arg}} , the filter filter\_f would be passed to the variable var and argument as arg.

create a filter to modify a string based on the argument. Create a simple view in views.py, that renders a string as follows:

1def my\_view(request):

2 context = {

3 "author": "satish gupta",

4 }

5 return render(request, "index.html", context)

python

After creating views, create a simple template filter named modify\_name in custom\_tags.py.

1from django import template

2

3register = template.Library()

4

5def modify\_name(value, arg):

6 # if arg is first\_name: return the first string before space

7 if arg == "first\_name":

8 return value.split(" ")[0]

9 # if arg is last\_name: return the last string before space

10 if arg == "last\_name":

11 return value.split(" ")[-1]

12 # if arg is title\_case: return the title case of the string

13 if arg == "title\_case":

14 return value.title()

15 return value

16

17register.filter('modify\_name', modify\_name)

python

You can now use the modify\_name filter in your templates.

1{% comment %} index.html {% endcomment %}

2

3{% load custom\_tags %}

4

5{{ author | modify\_name:"first\_name"}}<br>

6{{ author | modify\_name:"last\_name"}}<br>

7{{ author | modify\_name:"title\_case"}}<br>

8{{ author | modify\_name:"first\_name" | modify\_name:"title\_case"}}<br>

## Writing Custom Template Tags

Template tags are more powerful and more complex than filters. Django supports several shortcuts for making most of them easier.

### simple\_tag

simple\_tag takes several arguments and returns a result after doing some processing. Below is a simple example to display the current date and time in the template

1# custom\_tags.py

2from django import template

3register = template.Library()

4

5import datetime

6

7@register.simple\_tag

8def current\_time(format\_string):

9 return datetime.datetime.now().strftime(format\_string)

python

Next, use the above custom template tag into your template as follows:

1{% comment %} index.html {% endcomment %}

2

3{% load custom\_tags %}

4

5{% current\_time "%d/%m/%Y %H:%M:%S %p" %}

django

This will give you the current date and time when you run the Django server.

117/11/2020 14:15:59 PM

### inclusion\_tag:

inclusion\_tag is a common type of template tag, and helps in displaying data by rendering another template. This tag is useful in scenarios when you want to render data that is common in several pages.

In the example below, the Users table is rendered by creating a new template named users.html using the show\_users\_table template tag. The custom\_tag.py is modified as given below:

1# custom\_tags.py

2from django.contrib.auth import get\_user\_model

3from django.template.loader import get\_template

4

5from django import template

6register = template.Library()

7

8User = get\_user\_model()

9

10def show\_users\_table():

11 users = User.objects.all()

12 return {'users': users}

13

14users\_template = get\_template('users.html')

15register.inclusion\_tag(users\_template)(show\_users\_table)

python

Next, create a new template named users.html and render the users in a tabular format.

1{% comment %} users.html {% endcomment %}

2

3{% load custom\_tags %}

4

5<table>

6 <thead>

7 <tr>

8 <td>Username</td>

9 <td>First name</td>

10 <td>Last name</td>

11 <td>email</td>

12 </tr>

13 </thead>

14 <tbody>

15 {% for user in users %}

16 <tr>

17 <td>{{user.username}}</td>

18 <td>{{user.first\_name}}</td>

19 <td>{{user.last\_name}}</td>

20 <td>{{user.email}}</td>

21 </tr>

22 {% endfor %}

23 </tbody>

24</table>

django

Now, you can use the above inclusion tag in any other template (say index.html) as shown below.

1{% comment %} index.html {% endcomment %}

2

3{% load custom\_tags %}

4

5{% show\_users\_table %}

django

This will render the user in a tabular manner