Chapter-7

Working with Advanced Template Features

Advanced Templates:

1) Template Inheritance
2) Template Filters

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 040 – 64 51 27 86, 80 96 96 96 96, 92 46 21 21 43 | www.durgasoft.com

3) Template tags for relative URLs

1) Template Inheritance:

- If multiple template files have some common code,it is not recommended to write that common code in every template html file. It increases length of the code and reduces readability. It also increases development time.
- We have to seperate that common code into a new template file,which is also known as base template. The remaining template files should required to extend base template so that the common code will be inherited automatically.
- Inheriting common code from base template to remaining templates is nothing but template inheritance.

How to implement Template Inheritane:

base.html

```
1)  <!DOCTYPE html>
2)   html,css,bootstrap links
3)   <body>
4)     common code required for every child tempalte
5)     {% block child_block%}
6)       Anything outside of this block available to child tag.
7)       in child template the specific code should be in this block
8)     {%endblock%}
9)   </body>
10) </html>
```

child.html

```
1)  <!DOCTYPE html>
2)  {%extends 'testapp/base.html'%}
3)    {% block child_block %}
4)       child specific extra code
5)    {%endblock%}
```

```
1)  body{
2)    background: url(https://images.unsplash.com/photo-1460650671472-
    ba449b1a6f10?ixlib=rb-
```


```
    0.3.5&ixid=eyJhcHBfaWQiOjEyMDd9&s=daeb797a01124d5e37bdb0f543314f90&au
    to=format&fit=crop&w=666&q=80);
3)    background-repeat: no-repeat;
4)    background-size: cover;
5)
6)  }
7)  .navbar{
8)    background:orange;
9)  }
10) .navbar a{
11)   color:white;
12) }
13) img{
14)   height: 200px;
15)   width: 30%;
16)   margin:1.5%;
17)   border:2px orange solid;
18) }
19) #home{
20)   font-size: 100px;
21)   font-weight: 900;
22)   color:white;
23)   text-align: center;
24) }
25) h1{
26)   font-size: 40px;
27)   font-weight: 900;
28)   color:white;
29)   text-align: center;
30) }
31) li{
32)   color:yellow;
33)   font-size: 30px;
34) }
```

Advantages of Template Inheritance:

1) What ever code available in base template is by default available to child templates and we are not required to write again.Hence it promotes Code Reusability (DRY).
2) It reduces length of the code and improves readability.
3) It reduces development time.
4) It provides unique and same look and feel for total web application.

Note: Based on our requirement we can extend any number of base templates.i.e Multiple Inheritance is applicable for templates.

How to Add Seperate CSS Files to Child Templates?

base.html

```
1)  <!DOCTYPE html>
2)  {%load staticfiles %}
3)  <html lang="en" dir="ltr">
4)    <head>
5)      <meta charset="utf-8">
6)      <title></title>
7)      <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7
    /css/bootstrap.min.css" integrity="sha384-
    BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
    crossorigin="anonymous">
8)      <link rel="stylesheet" href="{% static "css/advtempot.css"%}">
9)      {% block xyz %}{% endblock %}
10)   </head>
11)   <body>
12)    <nav class="navbar navbar-default navbar-inverse">
13)   <div class="container">
14)    <!-- Brand and toggle get grouped for better mobile display -->
15)    <div class="navbar-header">
16)      <a class="navbar-brand" href="/">DURGA NEWS</a>
17)    </div>
18)      <ul class="nav navbar-nav">
19)        <li><a href="/">Home</a></li>
20)        <li><a href="/movies">Movies</a></li>
21)        <li><a href="/sports">Sports</a></li>
22)        <li><a href="/politics">Politics</a></li>
23)      </ul>
24)   </div><!-- /.container-fluid -->
25) </nav>
26) {% block  body_block  %}
27) {%  endblock  %}
28)   </body>
29) </html>
```

child.html

```
1)  <!DOCTYPE html>
2)  {% extends 'testapp/base.html'%}
```

```
3)  {%load staticfiles%}
4)  {% block xyz  %}
5)  <link rel="stylesheet" href="{%static "css/new.css" %}">
6)  {%endblock%}
7)    {% block  body_block  %}
8)     <h1>Welcome to DURGA NEWS PORTAL</h1>
9)    {%  endblock  %}
```

2) Tempalte Filters:

- In the template file, the injected data can be displayed by using template tags.
- {{emp.eno}}
- Before displaying to the end user if we want to perform some modification to the injected text, like cut some information or converting to title case etc, then we should go for Template filters.

Syntax of Template Filter: {{value|filtername:"argument"}}

Filter may take or may not take arguments.i.e arguments are optional.

Eg:

```
<li>{{msg1|lower}}</li>
```

msg1 will be displayed in lower case

```
<li>{{msg3|add:"Durga"}}</li>
```

"Durga" will be added to msg3 and then display the result to the end user

```
{{ msg|title }}
{{ my_date|date:"Y-m-d" }}
```

Note: There are tons of built in filters are available.
https://docs.djangoproject.com/en/2.1/ref/templates/builtins/#ref-templates-builtins-filters

How to Create our own Filter:

Based on our requirement we can create our own filter.

Steps:

1) Create a folder 'templatetags' inside our application folder.
2) Create a special file named with __init__.py inside templatetags folder, so that Django will consider this folder as a valid python package
3) Create a python file inside templatetags folder to define our own filters
   cust_filters.py -->any name

cust_filters.py

```
1)  from django import template
2)  register=template.Library()
3)
4)  def first_eight_upper(value):
5)      '''This is my own filter'''
6)      result=value[:8].upper()
7)      return result
8)
9)  register.filter('f8upper',first_eight_upper)
```

Note: We can also register filter with the decorator as follows.

```
1)  from django import template
2)  register=template.Library()
3)
4)  @register.filter(name='f8upper')
5)  def first_eight_upper(value):
6)      '''This is my own filter'''
7)      result=value[:8].upper()
8)      return result
```

f8upper is the name of the filter which can be used inside template file.

4) Inside template file we have to load the filter file as follows(In the child template bunot in base template) `{%load cust_filters%}`

5) We can invoke the filter as follows `{{msg|f8upper}}`

movies.html

```
1)  <!DOCTYPE html>
2)  {%extends 'testapp/base.html'%}
3)   {% block body_block %}
```

```
4)    <h1>Movies Information</h1><hr>
5)    {%load cust_filters%}
6)    <ul>
7)      <li>{{msg1|lower}}</li>
8)      <li>{{msg2|upper}}</li>
9)      <li>{{msg3|add:"--Durga"}}</li>
10)     <li>{{msg4|f8upper}}</li>
11)     <li>{{msg5}}</li>
12)   </ul>
13)   {%endblock%}
```

Eg 2: Custom Filter with argument

```
@register.filter(name='c_and_c')
def cut_and_concate(value,arg):
   result=value[:4]+str(arg)
   return result
```

Note: The main advantage of template filters is we can display the same data in different styles based on our requirement.

Example-2

cust_filters.py

```
1)  from django import template
2)  register=template.Library()
3)
4)  @register.filter(name='truncate5')
5)  def truncate5(value):
6)     result=value[0:5]
7)     return result
8)
9)  @register.filter(name='t_n')
10) def truncate_n(value,n):
11)    result=value[0:n]
12)    return 'durga'
```

lower.html

```
1)  <!DOCTYPE html>
2)  <html lang="en" dir="ltr">
3)    <head>
```

```
4)    <meta charset="utf-8">
5)    <title></title>
6)   </head>
7)   <body>
8)    {%load cust_filters%}
9)    {% for r in data_list%}
10)  <ul>
11)    <li>{{r.name|t_n:6}}</li>
12)    <li>{{r.subject|t_n:3}}</li>
13)    <li>{{r.dept|t_n:2}}</li>
14)    <li>{{r.date|timesince}}</li>
15)    <li>{{r.date|date:"d-m-Y"}}</li>
16)  </ul><hr>
17)  {%endfor%}
18) </body>
19)</html>
```

3) Template Tags for URLs:

- Thank You
- Thank You
- Thank You
- Thank You