Theory: Template filters

13 minutes 0 / 5 problems solved

When preparing an HTML template, you can use Python objects or even some of their methods for your convenience. However, Python is not the main means of working with HTML pages, so you need to be familiar with other tools for manipulating data, such as template filters.

§1. Filters

You can use strings as arguments to the built-in `len` function in Python, but you cannot use functions with arguments in templates. To get the size of a string in a template, we can use DTL filters.

Filters are special operations that work like Python functions: they receive arguments and return the result of their processing. For example, we can use the `length` filter to get the size of a string.

To use `length` in a template, apply this sequence: *expression, pipe operator, filter*. Pass an expression to the filter and get the result.

```
{{ "Lorem ipsum" | length }}
```

It works not only with constants but with any of your variables.

§2. Filters with arguments

Sometimes we want to make simple math calculations, for example, find the sum of two numbers. Bad news is, you cannot do math in templates with the operators `+` and `-` like in Python. Good news is, we can use filters with arguments for this purpose!

To add two numbers in a template, add a colon after the filter and pass the second argument to it:

```
{{ 100000 | add:500 }}
```

Don't forget to use a colon! If you omit it, the template won't work.

As you see, the syntax is not that different, and there's nothing scary about additional arguments.

> Only some filters have arguments. Please refer to the documentation to prevent mistakes in your code and see what other filters Django has to offer.

§3. Pipelines

One filter is fine but sometimes we need to combine them, just like calling a function from another function. In such situations, we can use pipelines.

Current topic:

Template filters Stage 2

Topic depends on:

✕ Django template language Stage 2

Table of contents:

Pipelines take their name from the use of pipe operators that pass the result of the filter processing *through a pipe* to the next filter. Using two, three, or four filters isn't too different from using just one because the syntax stays the same, and the outcome of the previous step becomes the input in the next.

For example, say we want to show the count of page views to our users. Assuming that something bad may happen and we forget to pass the necessary variable, we use the `default` value to show at least 0 views and thus make the layout of the page consistent.

```
{{ counter | default:0 | add:1 }}
```

We also add 1 to the counter to account for the current view of the page.

§4. Example

John Doe asked us again to redesign his blog. He wants to see some extra information for every post he made. Here is what John's first post looked like:

```
import datetime

post = {
    "created_at": datetime.datetime(2019, 5, 20, 14, 15, 43),
    "text": "My first post",
    "comments": [
        "My congratulations!!",
        "Looking forward to the second one",
    ],

}
```

Let's count the number of words in it, display when the record was created, and how many likes it has now:

```
<html>
  <body>
    <div>{{ post.text }}</div>
    <div>There are {{ post.text.split | length }} words in a post</div>
    <div>Created at: {{ post.created_at | date:"Y.m.d" }}</div>
    <div>Likes: {{ post.likes | default:0 | add:1 }}</div>
    <div>{{ post.comments | join:"<br />" }}</div>
  </body>
</html>
```

Notice how even though we forget to pass likes in the initial structure of a post, the `default` filter provides us the zero value. We think that John himself likes his post, and we apply the `add` filter to add one more like to it.

Check the documentation for the information on the date and join filters.

§5. Conclusion

Let's recap: in this topic, we learned what template filters are and how we can use them. Filters are just like Python functions: some of them have arguments, others don't, and just like with functions, we can combine them to get the final result.

Report a typo

54 users liked this theory. **1** didn't like it. **What about you?**

Start practicing

Comments (2) Hints (0) Useful links (0) Show discussion