Custom tags and filters

Let's create a Django app called `custom_template_tags` to demonstrate the usage of custom tags and filters.

Step 1: Create a Django App

First, create a Django app named `custom_template_tags`:

```
python manage.py startapp custom_template_tags
```

Step 2: Define Custom Tags

In the `custom_template_tags` app, create a Python module called `templatetags` if it doesn't already exist, and then create two Python files inside it: `custom_tags.py` and `custom_filters.py`.

custom_tags.py

```
# custom_template_tags/templatetags/custom_tags.py
from django import template
import random

register = template.Library()

@register.simple_tag
def random_number():
   return random.randint(1, 100)
```

custom_filters.py

```
# custom_template_tags/templatetags/custom_filters.py
from django import template

register = template.Library()

@register.filter
def to_upper(value):
   return value.upper()
```

Step 3: Register Custom Tags and Filters

Make sure to add an empty `__init__.py` file in the `templatetags` directory to make it a package. Then, load and use the custom tags and filters in your templates.

Step 4: Use Custom Tags and Filters in Templates

Now, you can use the custom tags and filters in your Django templates. Here's an example template (`example_template.html`):

```
<!-- example_template.html -->
{% load custom_tags custom_filters %}

<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Custom Tags and Filters Example</title>
</head>
<body>
   <h1>Random Number: {% random_number %}</h1>
   <h2>Uppercase Text: {{ text|to_upper }}</h2>
</body>
</html>
```

Step 5: Configure URLs and Views (Optional)

If you want to render this template from a view, you can create a view and configure the URLs accordingly.

Step 6: Test Your Custom Tags and Filters

Finally, run your Django app and navigate to a page where the template is rendered. You should see the output of the custom tags and filters in action.

That's it! You've now created a Django app that demonstrates the usage of custom tags and filters. These custom tags and filters can be used to add custom logic and functionality to your Django templates, enhancing their capabilities and making them more powerful and versatile.