Theory: Template tags

16 minutes 0 / 5 problems solved

One of the basic principles in programming is to reuse code that you already have. There is a good chance that we'll have some identical parts of code for different pages, for example, *header*, *footer*, or *contacts.* It makes sense to define them once and then reuse them elsewhere. This is one of the things that Django tags can help us with. Let's learn more about where and how we can use them!

§1. Include

Tags are special constructions embraced in the operators `{%` and `%}`. They can help you process parts of your template or include external resources like predefined HTML code or even other templates. You should be already familiar with the `{% if %}` and `{% for %}` tags.

> Some tags like `{% if %}` and `{% endif %}` need opening and closing parts to work correctly, others don't. Please refer to the documentation for more information about tags.

Let's take a look at specific examples. Our blogger friend John Doe needs some help with his blog: he wants to show his contacts in each post. We create a simple template in the file *blog/templates/blog/contacts.html* for the *contacts* section on a page:

```
<table>
  <tr>
    <td>Phone:</td>
    <td>0-123-456-7890</td>
  </tr>
  <tr>
    <td>Email:</td>
    <td>john@doe.com</td>
  <tr>

</table>
```

To include it in any other template, we can use the `{% include %}` tag:

```
<!-- some content -->
{% include "blog/contacts.html" %}
<!-- some content -->
```

This way, we don't need to copy the same code with contacts over and over again: we just include it. If we ever want to change what the *contacts* look like, all we need is to change the *contacts.html* file, and all other templates will have the updated *contacts* right away.

§2. Extend

You know how to include predefined parts to create a new template. Once you start using this technique, you'll probably notice that the includes are repeating across different templates. Our goal was to minimize repetition, but it seems that it's still there. To solve this problem, let's define the base template, extend it, and populate only the parts that are different.

As a rule, all HTML pages for a particular site have a lot in common. If you don't want to repeat the same code over and over, you can slice your template into several **blocks** and then redefine only the blocks you need.

Let's make the base template in the file *blog/templates/blog/base.html* with the `title` and `content` blocks in it:

Current topic:
Template tags Stage 3

Topic depends on:
Django template language Stage 2

Topic is required for:
Static content Stage 5

Table of contents:
↑ Template tags
§1. Include
§2. Extend
§3. CSRF token
§4. Conclusion
Feedback & Comments

```
<html>
  <head>
    <title>{% block title %} John Doe's Blog {% endblock %}</title>
  </head>
  <body>
    {% include "blog/contacts.html" %}
    {% block content %} Hello {% endblock %}
  </body>
</html>
```

Now you can extend a custom page from the base. To make one, add `{% extends "base.html" %}` at the beginning of your file:

```
{% extends "base.html" %}
{% block content %} {{ block.super }} world! {% endblock %}
```

If you want to extend the content of the default block, you can paste the `{{ block.super }}` variable and then add new data to the block. The blocks that you didn't redefine will simply stay the same. This way, you can make a new page for the site with just a few lines without having to copy the entire HTML layout.

§3. CSRF token

To send data to the server, we use HTML forms. Through forms, users can send confidential data and make financial transactions, so we should secure the forms from potential spoofing.

Let's create a template to add comments to posts. Saving comments in handlers is too much for now, so let's just prepare a section that we'll include later when we learn how to process requests on the server:

```
<form action="/comment/save" method="post">
  {% csrf_token %}
  <input name="text">
  <input type="submit" value="Save">
</form>
```

If you look closely at the code snippet, you'll surely notice a tag `{% csrf_token %}`. CSRF stands for Cross-Site Request Forgery. We don't want any fraud, so in forms, we must always use this tag to secure our applications. CSRF token is a generated sequence of symbols that the server uses to identify the user's session. If the sequence matches, the form is considered reliable.

> Including CSRF tokens in the POST requests is obligatory in Django by default. You can turn off the verification in your handlers, but it's highly recommended to keep it.

There are a lot more security issues that we can face in our web applications, but with the CSRF token we'll have one problem less.

§4. Conclusion

In this topic, we learned how to include one template into another and create a base template that can be partially extended. These tools are going to be very helpful as your project grows. Don't forget about security: add CSRF tokens in forms to secure sessions for our users.

Report a typo

52 users liked this theory. **2** didn't like it. **What about you?**

Start practicing

Comments (3) Hints (0) Useful links (1) Show discussion