Advanced Programming with Python. Session 4

Pepe García

2020-04-20

Advanced Programming with Python. Session 4

Plan for today

- Learn more about HTML
- Using HTML templates

HTML on Flask

So far we have seen only a very simple setup of HTML, let's create a website with several pages.

We'll have a **homepage** and an **about** page

HTML on Flask

Let's see **example-1** in the **session-4** repo

HTML on Flask

Let's modify our example and add a link to the **about** page in all pages.

Templates

Templates in Flask provide:

- Separation of concerns
- Code reuse
- A nice way of creating HTML interfaces

Templates

Templates look a lot like normal HTML, but they provide some special markup.

```
<html>
  <head>
    <title>Index</title>
  </head>
  <body>
    <h1>Hello {{name}}</h1>
  </body>
</html>
```

Templates

We can render templates using the **render_template** function, from **flask**

```
from flask import render_template

@app.route("/index")
def index():
    return render_template(
        "index.html",
        name="Pepe")
```

We can pass variables to the **render_template** function that will be available inside the template!

HTML on Flask

See **example-2** in the **session-4** repo

Control statements. if

Apart from the **{{double_brackets}}**, that will be substituted with the corresponding value, flask templates also support control statements using **{% %}** blocks.

```
{% if name %}
  <p>the name was {{name}}</p>
{% else %}
  <p>We didn't receive any name</p>
{% endif %}
```

Control statements. **for**

We will also be able to iterate over a sequence of values using the **{% for ... %}** block!

```
{% for name in names %}
  <p>the name was {{name}}</p>
{% endfor %}
```

Practice

Do the **exercise-1** in the repository, you'll need to render all tweets in the **tweets** list in the tweets template!

Interlude: **Some more HTML**

HTML. **divs**

We'll use **div** as a generic container in HTML. They will take all the width available (**display: block**).

```
<div>
  <p>HTML is such a cool language</p>
  <p>We can group things inside div elements</p>
</div>
```

HTML. **spans**

We'll use **span** as a generic container in HTML. They will stack one beside the next one. (**display: inline**).

```
<span><img src="https://google.com/logo.png"/></span>
<span>you can see Google's logo to the left</span>
```

HTML. **ul**

We'll use **ul** for representing unordered lists in HTML. Each one of the elements of the list will be created using the **li** tag.

```
<div>
  There are four members in The Beatles:

  <ul>
    <li>Ringo</li>
    <li>John</li>
    <li>Paul</li>
    <li>George</li>
  </ul>
</div>
```

Template Inheritance

In most websites we're going to have some parts of them that are repeated, such as the navigation menu, the footer, etc.

We will use template inheritance to not repeat ourselves.

Template inheritance

We will start by creating a base template that has all the common parts of our website.

```
<html>
  <head>
    <title>{{title}}</title>
  </head>
  <body>
    <!-- menu -->
    <a href="http://localhost:5000/"><h1>My website</h1></a>

    {% block main %}{% endblock %}

    <!-- footer -->
    <p>All rights reserved :)</p>
  </body>
```

Template inheritance

After we've created our base template we can extend it from others!

```
{% extends "base.html" %}

{% block main %}
<p>
  this is specific to this template, not inherited from the parent one!
</p>
{% endblock %}
```

Practice

Using template inheritance, fix the **example-1** so that we don't repeat ourselves.