Advanced Programming with Python

HTML and Flask

Pepe García jgarciah@faculty.ie.edu

2020-04-20

Plan for today

- Review homework
- Learn more about HTML
- Using HTML templates

Homework

HTML Refresher

Let's refresh a couple of HTML elements learnt the last day

HTML. **div**

We'll use ** ** as a generic container in HTML.

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

** ** will take all the available width.

HTML. span

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

We'll use **** as a generic container in HTML.

They will stack besides the previous one.

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>
```

HTML on Flask

So far we have seen only a very simple setup of HTML,

Example 1

Let's create a website with several pages.
We'll have a **home.html** and an **about.html** page.

HTML on Flask

So far we have seen only a very simple setup of HTML,

Example 1

Let's create a website with several pages.
We'll have a **home.html** and an **about.html** page.
And now let's modify our website and add a footer to all pages.

HTML on Flask

Question

What problems do you see whith this workflow?

HTML on Flask

Question

What problems do you see whith this workflow?

- error prone

HTML on Flask

Question

What problems do you see whith this workflow?

- error prone
- repetitive

HTML on Flask

Question

What problems do you see whith this workflow?

- error prone
- repetitive
- tedious

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

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

```
<html>
  <head>
    <title>Index</title>
  </head>
  <body>
    <h1>Hello {{name}}</h1>
    <!-- {{ }} are used to interpolate variables -->
  </body>
</html>
```

Templates

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

```
from flask import render_template

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

Variables for the template are passed as arguments to the **render_template** function. In this case `name="Pepe"`.

Templates

Example 2

See **`example2.py`** in the repository for a full example

Control statements. `if`

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

We can use the **if** keyword in templates as we would do in Python.

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

Control statements. `if`

Exercise 1

We have this **`logged_in`** variable in our `exercise1.py` server. Depending on the value it has, we want to print either

```
Welcome to the private area
```

or

```
You're not logged in, get out!
```

Control statements. `for`

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

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

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

Control statements. `for`

Exercise 2

In `exercise2.py` we have a list of members of the beatles, make sure that you show them in the webpage.

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.

Template inheritance can help us get rid of repeated code and clean things up.

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

Template inheritance

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

    <main>
    {% block main %}{% endblock %}  <!-- main -->
    </main>

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

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 %}
```

Template inheritance

Exercise 3

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

Recap

- Repeating HTML code has downsides, we should avoid it as much as possible

Recap

- Repeating HTML code has downsides, we should avoid it as much as possible
- Flask templates can interpolate variables with **{{}}**

Recap

- Repeating HTML code has downsides, we should avoid it as much as possible
- Flask templates can interpolate variables with `{{}}`
- Flask templates can use control flow operators such as `{% if ... %}` and `{% for ... %}`

Recap

- Repeating HTML code has downsides, we should avoid it as much as possible
- Flask templates can interpolate variables with `{{}}`
- Flask templates can use control flow operators such as `{% if ... %}` and `{% for ... %}`
- Flask templates extend other templates and only overwrite needed `blocks`