Advanced Programming with Python

Forms in HTML and Flask

Pepe García jgarciah@faculty.ie.edu

2020-04-20

Plan for today

- Errata from last day: HTML escaping, rich text vs plain text
- HTML forms
- handling HTML forms in flask
- Time for the individual assignment

HTML escaping

Remember how we escaped characters in Python when, for example we wanted to use the double quote inside a string?

```
value = "\"like this, for example\""
```

HTML escaping

In HTML we will need something similar for lots of characters. We cannot directly write **<**, for example, and expect it to be rendered whent he page gets rendered, since **<** is already part of the markup language.

We need to transform it to HTML entities:

HTML escaping

In HTML we will need something similar for lots of characters. We cannot directly write **<**, for example, and expect it to be rendered whent he page gets rendered, since **<** is already part of the markup language.

We need to transform it to HTML entities:

character	HTML Entity
"	&quot;
'	&apos;
&	&amp;
<	&lt;
>	&gt;
é	&eacute;
à	&agrave;

HTML escaping

Rich text vs Plain text

Rich text editors, such as **TextEdit.app** in Mac, will convert the values we write to HTML entities directly.

We need to use a plain text editor instead (Spyder, VSCode, Emacs...)

HTML forms

Whenever we want to gather data from the user in HTML, we'll use forms.

HTML forms

Whenever we want to gather data from the user in HTML, we'll use forms.

All fields in forms must be inside a **<form>** tag

HTML forms

Whenever we want to gather data from the user in HTML, we'll use forms.

All fields in forms must be inside a **<form>** tag

```
<form
  action="http://localhost:5000/form"
  method="POST">
 ...
</form>
```

HTML forms

Whenever we want to gather data from the user in HTML, we'll use forms.

All fields in forms must be inside a **<form>** tag

```
<form
  action="http://localhost:5000/form"
  method="POST">
 ...
</form>
```

We'll put the URL for handling the form in the **action** attribute.

HTML forms

Whenever we want to gather data from the user in HTML, we'll use forms.

All fields in forms must be inside a **<form>** tag

```
<form
  action="http://localhost:5000/form"
  method="POST">
 ...
</form>
```

We'll put the URL for handling the form in the **action** attribute.

And the HTTP method in the **method** attribute

HTML forms. Fields

<input> are used for declaring different kinds of inputs from the user.

HTML forms. Fields

<input> are used for declaring different kinds of inputs from the user.

We'll always need to give a unique **name** to it and a **type**

```
<input name="user" type="text">
```

HTML forms. Fields

There are a lot of types of inputs we can use.

HTML forms. Fields

There are a lot of types of inputs we can use.

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input

```
<input name="pass" type="password">
<input name="date" type="datetime-local">
...
```

HTML forms. Submit

In order to create a button that submits the **form**, we'll use

```
<input type="submit" value="send the form!">
```

HTML forms

Exercise

Create a simple login form in HTML. password field, and a submit button.

Handling HTML forms in flask

We can access data from the <form> using the **request** object in Flask:

```
from flask import request, jsonify

@app.route("/handle", methods = ["POST"])
def handle_form_submission():
    return jsonify(request.form)
```

Handling HTML forms in flask

We can access data from the <form> using the **request** object in Flask:

```
from flask import request, jsonify

@app.route("/handle", methods = ["POST"])
def handle_form_submission():
    return jsonify(request.form)
```

the keys in the **form** dictionary are the values we put in the **name** attribute of the **<input>**

Differences between GET and POST in forms

The big difference between them is that, when selecting `GET`, the data will be sent as query parameters, while when selecting `POST`, it will be sent in the request body

Handling HTML forms in flask

Exercise

Create a login form that checks if the `user` and `password` sent by the user exist in the database.
In case they exist, render the `private.html` template,
Otherwise, render the `unauthorized.html` template with a 401 unauthorized method

Recap

- We'll gather data from the user in the front side with HTML `<form>`
- `<input>` comes in several flavours: `type="password"`, `type="text"`, `type="email"`...
- From the server side, we'll receive the contents of the form in the `request.form` dictionary