Advanced Programming with Python

Forms in HTML and Flask

Pepe García jgarciah@faculty.ie.edu

Plan for today

- HTML forms
- Handling HTML forms in flask
- Time for QA

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

```html
<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`**

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

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`

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

```html
<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

```html
<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:

```python
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:

```python
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 (in the URL), while when selecting **`POST`**, it will be sent in the request body

Handling HTML forms in flask

Exercise - paymepal

In this exercise we'll see how can we handle user input via forms.

Handling HTML forms in flask

Exercise - paymepal

In this exercise we'll see how can we handle user input via forms.

- in `index.html`: create a login form that contains two fields, a `user` and `password`, and a `submit` input too.

Handling HTML forms in flask

Exercise - paymepal

In this exercise we'll see how can we handle user input via forms.

- in `index.html`: create a login form that contains two fields, a `user` and `password`, and a `submit` input too.
- in `paymepal.py`: check if the user exists in the users dictionary, from the data module. If the user exists, render `private.html`, and `unauthorized.html` otherwise.

Handling HTML forms in flask

Exercise - paymepal

In this exercise we'll see how can we handle user input via forms.

- in `index.html`: create a login form that contains two fields, a `user` and `password`, and a `submit` input too.
- in `paymepal.py`: check if the user exists in the users dictionary, from the data module. If the user exists, render `private.html`, and `unauthorized.html` otherwise.
- in `paymepal.py`: get the transactions from the user if there is any, and render them in `private.html`.

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