Data Handling Using the Request Object

In this lesson, we will handle the POST request received by the login template and retrieve the data from it.

Accessing form data Getting the Method type from request Explanation Getting Form data from request Explanation Complete implementation Explanation In app.py In login.html

In the last lesson, we were sending both the GET and POST requests to the login view function. However, there was no differentiation between these requests inside the view.

So, how can we know if a particular request is GET or POST?

This is where the request object comes in handy. Let's find out more about the request object.

Accessing form data

To access the data sent by a user, we use the global request object. Before we start using it, let's import it from the flask module.

from flask import request

Getting the Method type from request

We can use the method member variable of the request object to determine the method of an incoming request.

Consider the snippet given below.

```
@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        ...
    else
        ...
    return render_template("login.html")
```

Explanation

In the snippet given above, at **line #3**, we check if the method is **POST**. Then, the logic for handling a **POST** request can be implemented at **line #4**. Additionally, if the **method** is anything other than **POST**, then the logic for that can be added at **line #6**.

Getting Form data from request

We can use the form member variable of the request object to obtain values that the user submitted. The form variable is a special data structure called an ImmutableMultiDict. But don't get confused by it. You can still use it easily. Consider the snippet given below.

```
@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        email = request.form["email"]
        password = request.form["password"]
        ...
    else:
        ...
    return render_template("login.html")
```

Explanation

In the snippet above, at **line** #4 - 5, you can observe the syntax for extracting the **form** data. Notice that the **key** for the **form** dictionary corresponds to the name attribute of the **input** tag. An alternate method can be:

```
if request.method == "POST":
```

```
email = request.form.get("email")

password = request.form.get("password")
...
```

Both of these syntaxes are valid.

Complete implementation

Now, let's wrap all the information into a working example. For the sake of this demo, consider that we possess a record of user credentials stored in the form of a dictionary. The application given below provides a straightforward user validation logic.

```
#header {
  padding: 30px;
  text-align: center;
  background: #140005;
  color: white;
  font-size: 40px;
}
#footer {
   position: fixed;
  width: 100%;
  background-color: #BBC4C2;
  color: white;
   text-align: center;
   left: 0;
   bottom:0;
}
ul {
 list-style-type: none;
 margin: 0;
  padding: 0;
}
  display: inline;
```

Explanation

Let's break down the changes we made in the application.

```
In app.py #
```

- On **line** #5 8, we are provided with a dictionary containing valid usernames and passwords.
- For the validation, we use an if condition in line 19 to check if email

Aloto as a Rey III the alchoral y associated with the value of password

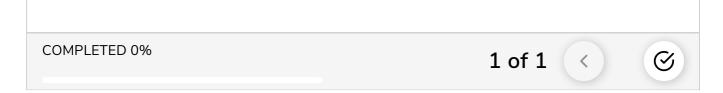
- If the comparison is successful, then the user is validated and, **in line #20**, the same template is rendered again with a message variable indicating the validation.
- In case the username and password provided were incorrect, then the same template is returned, and an error message is sent using the message variable in line #21.

In login.html

- To show the message inside the template, we added a conditional statement at line #9. This statement checks if a message variable was received.
- Next, if the message variable was indeed received, then in line #10, its value is shown to the user.

Quick Quiz!

Which of these if statements can determine if a POST request was recieved?



Now that we understand how to handle forms using the request object, in the next lesson, we will discuss the other method. Stay tuned!