Introduction to Forms and Requests

In this lesson, we will learn about the two methods of handling forms in Flask.

WE'LL COVER THE FOLLOWING

Handling forms in Flask

The `Flask` package itself does not provide us a way to handle forms. There are two ways that developers deal with this:

1. **Via the `request` object**
2. **Via the `Flask-WTF` extension**

In this lesson, we will briefly discuss the first option. Later, we will focus more on the second option because it is more popular and more comfortable to maintain for broad applications.

Using the request object

Here, we will take the example of a simple **login** form. We will be creating the template and route for this form in this lesson. In the next lesson, we will be using the `request` object for data handling. Let's get started.

Create `HTML` templates

Consider a simple login template.

```
<h1>Login</h1>
<form method="POST">
  Email:<br>
  <input type="text" name="email" >
  <br>
  Password:<br>
  <input type="password" name="password" >
  <br><br>
  <input type="submit" value="Login">
</form>
```

In the template given above, we created a simple form with `email` and `password` fields. A `submit` button was also added to submit the login form. Now, let's connect this form to a new login route.

Create a new route for login

Let's modify the example we used in the last chapter.

1. Add a `/login` route in the application.
2. Create `login.html` and extend it from `base.html`.

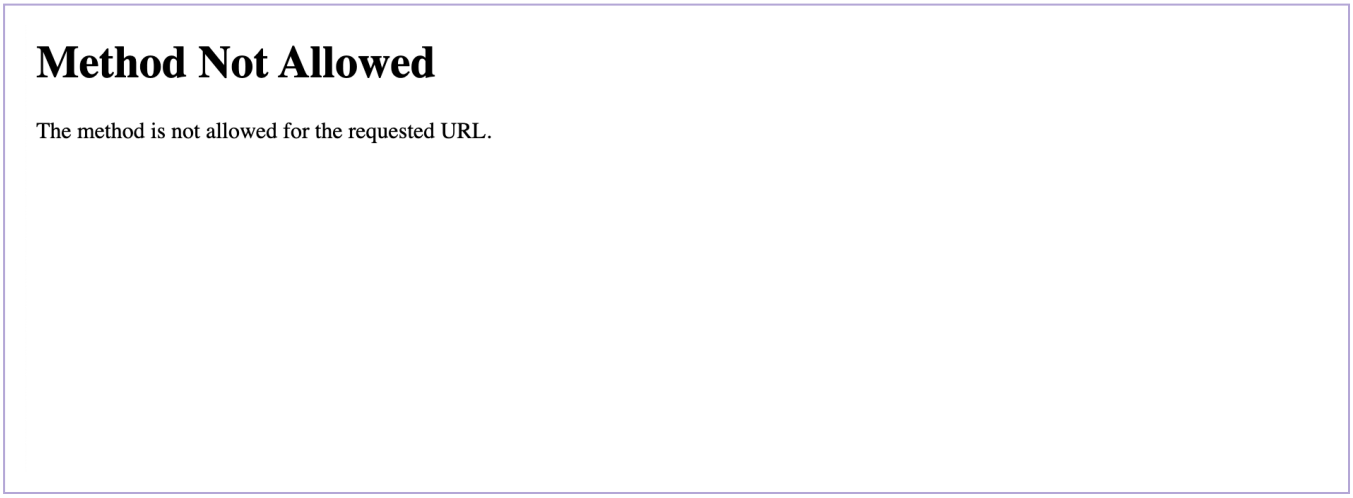
```
#header {
  padding: 30px;
  text-align: center;
  background: #140005;
  color: white;
  font-size: 40px;
}
#footer {
   position: fixed;
   width: 100%;
   background-color: #BBC4C2;
   color: white;
   text-align: center;
   left: 0;
   bottom:0;
}
ul {
  list-style-type: none;
  margin: 0;
 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

li {
  display: inline;
}
```

> 📌 **Note:** At **line #9**, inside the `form` tag we have also added an `action` field equal to `{{url_for('login')}}`. This will direct the submit request to the `login` view function.

Add `POST` and `GET` methods

> 💡 **Observe:** When you press the `submit` button on the log-in page, an error message appears.

This message is as follows:



The reason for this message is that the `login` view can not handle the `POST` request which we just sent from the `form`. On default, the `route()` decorator serves only the `GET` requests. Therefore, we have to provide an extra parameter, called `methods`, to the `route()` decorator (for the `login` view).

```
@app.route("/login", methods=["GET", "POST"])
```

The `method` argument is a Python list containing the names of allowed methods.

Complete implementation

After fixing the above-mentioned error, the complete program is given below.

```
#header {
  padding: 30px;
  text-align: center;
  background: #140005;
  color: white;
```

```
  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;
}

li {
  display: inline;
}
```

Now, let's move on to the next lesson where we will handle the data received from this form using the `request` object.