HTTP Methods (GET/POST) & Retrieving Form Data

In this flask tutorial I show you how to use the HTTP request methods Post and Get. The POST method will allow us to retrieve data from forms on our web page. In later videos we'll get into more advanced topics relating to login sessions and using POST methods to retrieve secure information like passwords.

Flask

HTTP Methods

In this tutorial we will talk about HTTP methods. HTTP methods are the standard way of sending information to and from a web server. To break it down, a website runs on a server or multiple servers and simple returns information to a client (web-browser). Information is exchanged between the client and the server using an HTTP protocol that has a few different methods. We will be discussing the post commonly used ones called POST & GET.

GET

GET is the most commonly used HTTP method and it is typically used to retrieve information from a web server.

POST

POST is commonly used to send information to web server. It is more often used when uploading a file, getting form data and sending sensitive data. POST is a secure way to send data to a web server.

Creating a Form

For this example will create a form and discuss how we can use the POST HTTP method to send the information in our form to our server.
Here is a new html file I created called "login.html".

login.html

```
{% extends "base.html" %}
{% block title %}Login Page{% endblock %}

{% block content %}
<form action="#" method="post">
    <p>Name:</p>
    <p><input type="text" name="nm" /></p>
    <p><input type="submit" value="submit"/></p>
</form>
{% endblock %}
```

We've specified the method of the form to be post. Which means that when the form is submitted we will POST data to the web server in the form of a request.
Note the name of the text input field, we will use this to get it's value from our python code.

Back-End

Create new python file called ***tutorial4.py***

What we're going to do is setup a login page that asks the user to type in their name. Once they hit the submit button they will be redirected to a page that displays the name they typed in.

We'll start by importing a new module called "request".

```
from flask import Flask, redirect, url_for, render_template, request
```

Next we'll setup the login page.

To specify that a page works with both POST and GET requests we need to add a method argument to the decorator.

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

Back-End

Now from inside the function we'll check if we are receiving a GET or POST request. We'll then define the operation to perform based on the request. If we have received a POST request that means we submitted the form and should redirect to the appropriate page. Otherwise we should simply return and render the login page.

```
def login():
    if request.method == "POST":
    user = request.form["nm"]
    return redirect(url_for("user", usr=user))
    else:
    return render_template("login.html")
```

Now we'll setup the user page. This will simply display the users name in h1 tags. Just for example/testing purposes, we don't need anything fancy.

```
@app.route("/<usr>")
def user(usr):
    return f"<h1>{usr}</h1>"
```

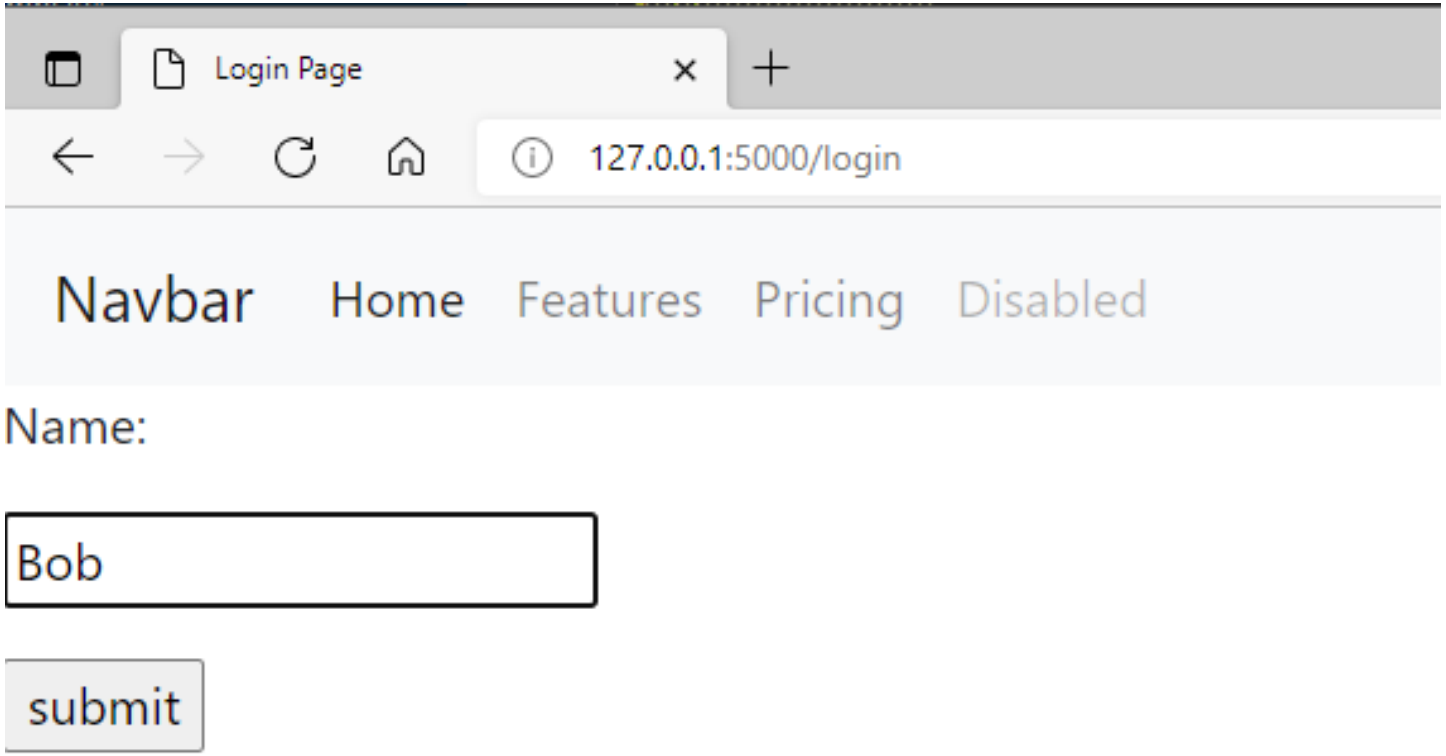
Full Python Code

tutorial4.py

```
from flask import Flask, redirect, url_for, render_template, request

app = Flask(__name__)

@app.route("/")
def home():
   return render_template("index.html")

@app.route("/login", methods=["POST", "GET"])
def login():
   if request.method == "POST":
      user = request.form["nm"]
      return redirect(url_for("user", usr=user))
   else:
      return render_template("login.html")

@app.route("/<usr>")
def user(usr):
   return f"<h1>{usr}</h1>"

if __name__ == "__main__":
   app.run(debug=True)
```

Now let's test out our site!

We'll head to the /login page and type our name.



Now let's test out our site!

And after hitting submit! We get a page with our name on it...