INSTRUCTIONS FOR LAB-3 FORMS

Lab 3-Forms   FlaskDemoLectures/03-Forms   Video 03-Forms

Copy the entire folder(project) 03-Forms. Rename it 03-Forms-Lab-*yourname*
Make sure that you have already installed flask-wtf (using pip install flask-wtf from within your Anaconda environment)

Part 1: Make a form in forms.py

Create a form to allow the user to enter additional snacks and calories.
In forms.py, copy and paste the LoginForm.
  Rename it to SnackForm, and then change the fields to input a snackName and calories.
  Our example only used boolean, password and string fields.
  You may want to look up an IntegerField and use that for the calories.
    Of course, you will have to import the IntegerField, as it is not currently imported
  Both fields should be required.
  When the user has completed the form, he/she should press a button that says "Add a Snack!"

Part 2: Make a route to handle this form, in flaskDemo.py

Copy the register route, paste it as a new route, and rename the route addSnack
  make changes as appropriate. (I changed "register" to "addSnack", and changed the flash message.)
  Clearly, we will have to add a template, named addSnack.html

At the top of the program, just under (or instead of) the posts=[list declaration,
  declare and assign values for your list of dictionaries.
  After all, posts are defined at the top of the file, so the home route recognizes them.
  Should we define the snack dictionary at the top of the file too? Or in the home route? Why?
  Otherwise, when the page first loads, it will have no idea what "FavSnacks" are.

```
favSnacks = list()
favSnack = dict()
favSnack['snack']="Nestle's Crunch"
favSnack['calories'] = 101
favSnacks.append(Snack)
favSnack=dict()
favSnack['snack']="KitKat"
favSnack['calories'] = 202
favSnacks.append(Snack)
```

This will initialize your list of favSnacks with favSnack dictionaires.
Alternatively, you can define it using the same format that the posts list of dictionaries uses.

Part 3: Create a template, addSnack.html

In Windows Explorer, or Mac Finder, copy register.html and name the copy addSnack.html
update the fields to reflect the input fields on snack form.
And, of course, you don't have to check if the user already has an account.

Part 4: Run the application.

Open Anaconda terminal. Navigate to the folder where your app is located.

Mac: Open Anaconda Navigator, then open a terminal, then navigate to folder where your app is located.

If you type in localhost:5000 for the url, you don't see anything about snacks.

Type in localhost:5000/addSnack

But that's kludgey....

Better: add a nav link.

In layout.html, you should add a nav link for addSnack

The url for the link is addSnack. But the link should display *Add a Snack*

Part 5: Fix the home page navigation to include a list of snacks.

And add the appropriate template for doing so.

home.html should display the snacks, not the posts that it currently displays.

First, change the home route in the flaskDemo.py file:

I will be creating a separate template named snacks.html. Or, you can just revise home.html.

Whichever you choose, make sure that render_template reflects the correct template.

For this project, sinceI named my template snacks.html, I used render_template snacks.html in my home route

Notice that it is expecting a title. So when you render the template, send it a title and also send it favSnacks.

Make sure that you are using favSnack and favSnacks instead of post and posts

in both your home route and also in your template.

Next, you have to change the template itself

In fact, you can completely replace the home.html page with lab.html from the 02-templates-lab-dict project

To refresh your memory, here is lab.html from that previous lab: (it will require some changes)

```
{% extends "layout.html" %}
{% block content %}
Here are some of {{title}}'s favorite snacks:
    {% for post in posts %}
        <article class="media content-section">
          <div class="media-body">

              <small class="text-muted">{{ post['snack'] }}</small>
                <small class="text-muted">{{ post.calories}}</small>
          </div>
        </article>
    {% endfor %}
{% endblock content %}
```

(Notice that I used the bracket notation once, and the dot notation once. Same difference.)

Notice that it is calling the list that was sent to it "posts".

Change "post" and "posts" to "favSnack" and "favSnacks". Just for readability.

Of course, that means you have to change things in the home route too!

You can call this template home.html. Or you can call it snacks.html. As long as you are consistent.

Make sure that it matches with what you wrote in render_template in your home route

Part 6: Add the new snack to the list of snacks.

Notice that when you run the app, the new snack added doesn't show up in the list.
That's because although we provided a form and a template (and more!),
We are displaying the entire list (snackList), but our new snack was never appended to the list.
We successfully completed the form, but never actually took the form fields and used them
to create another dictionary entry to add to the favSnacks dictionary!
And in our template, we are displaying the favSnacks dictionary.

Your job is to add the snack that was entered on the form, to the favSnacks dictionary.
That way, the next time that favSnacks are displayed, it will incude the new snack.
Look at how we originally created and added snacks to the dictionary.
Where would that have to be processed?
How do you extract the form fields? (Hint: look at the register route in flaskDemo.py…)