launch_code

LC101 2.10

Class Agenda

1. Last Class Review
2. Announcement (Geek Gala)
3. Guest Speaker
4. Studio

Announcement

GeekGala - LaunchCode's birthday party is happening on October 19th!

If you are interested in attending please refer to the announcement on canvas for more information.

Review

ORM - Object Relational Mapping

Creating a DB and configuring it within your Flask app

Create classes and map them to our DB within Flask

Working with an ORM

1. Create your DB, and a user for your DB. (last lecture)
2. Create a Flask app (last lecture)
3. Configure the DB in your Flask app (last lecture)
4. Create your classes that map to your DB (last lecture)
5. Create the logic in your route (today's lecture)
6. Create the templates (today's lecture)

Prep Work Walkthrough

Let’s continue our example from last week.

Adding Templates

Since we are creating a new User class, and Table to our DB let's start with our templates.

New users will need a place to register, and existing users will need a place to login.

Register.html

login.html

Create Routes

Now that we have some new .html files to render, let's create some routes for them.

```
@app.route(‘/register’, methods=[‘GET’]))

@app.route(‘/login’, methods=[‘GET’])
```

Task Class

We can add now access our DB from python using the terminal, and create new tasks, then committing the tasks to our DB.

We can now write some additional python code that will activate on a server call that will automatically write to our DB.

Expand routes

Now that we have the pages working correctly, we need to change some of our logic in main.py in order to handle the different methods coming our way.

The video did everything in the same route, I created a new route for each method, to keep everything nice and organized.

```
@app.route('/register', methods=['POST'])

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

Login and Register

On Login we need to verify if the user exists in the DB, and if the password entered matches the password in our DB.

On Register we need to check to see if the user does not exist already, and if they don’t we need to write them as a new user in our DB.

Sessions

Sessions give us a way to store a logged in user’s information efficiently.

To work with sessions we will first need to import it from Flask.

Session is a dictionary so we can write to session[‘email’] to save the user’s email while they are logged in to our application.

Then we can use it to check to make sure a user is logged in before letting them access our site.

before_request

Our prepwork introduced us to the concept of checking something before all HTTP requests are handled. Great for creating a ‘login wall’.

```
@app.before_request

Def require_login():
```

WhiteList

Our prep work introduced us to the concept of a whitelist. A whitelist is us specifying what pages can be accessed by people who are not logged in.

This is crucial, so that non logged in users can actually access the login page, to therefore login.

We do this by creating a list of allowed routes, and then changing the logic of our beore_request, to let them through on the allowed routes.

Secret Key

In order to allow your app to use sessions we must define a secret key in our Flask app configuration.

It’s just a random string, but your computer will have their browser verify it, to ensure their browser has access to your application, by knowing the secret_key.

App.secret_key = ‘SOMERANDOMSTRING’

Flash Messages

We need to give some messages to the user when they mistype, or somehow mess up with one of our forms.

Flash message - a tool built into flask that allows us to store, and pass messages to HTML from different locations.

This requires us to change our base.html template, and to add flash() messages when we want to render a change.

Relationships

Finally we need to go over how to link two of our tables together.

We need to add to our Model's for both User, and Task. For task we need to add a new column, that's a foreign key. For User, we need to set a backref to the Task column.

Guest Speaker

We are being joined by David Atchley!

He is the founder of Tandemly (http://tandem.ly/), and has over 20 years of experience as a developer!

He has found this success without a CS degree, and is a completely self taught programmer!

Studio - Flicklist 7

Walkthrough has us:

1. Creating a new user class
2. Adding routes
3. Adding a template
4. Editing our base.html template
5. Creating a login wall (using @app.before_request)
6. Flash messages from our app