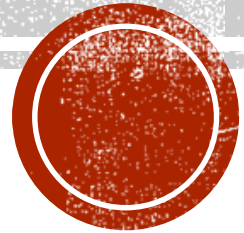
FLASK — DATABASE III



LAST TIME – WE ADDED A PROFILE

- Last time, we created a profile for a user

```
def profile():
    form = ProfileForm()
    if not session.get("USERNAME") is None:
        if form.validate_on_submit():
            cv_dir = Config.CV_UPLOAD_DIR
            file_obj = form.cv.data
            cv_filename = session.get("USERNAME") + '_CV.pdf'
            file_obj.save(os.path.join(cv_dir, cv_filename))
            flash('CV uploaded and saved')
            # now we add the object to the database
            user_in_db = User.query.filter(User.username == session.get("USERNAME")).first()
            profile = Profile(dob=form.dob.data, gender=form.gender.data, cv=cv_filename,user=user_in_db)
            db.session.add(profile)
            # remember to commit
            db.session.commit()
            return redirect(url_for('choice'))
```

NOW WE ADD REQUIREMENTS

- After login, the user must get a choice
  - Add a post
  - Update their profile

- When adding a post, user must:
  - Be able to see any previous posts made
  - If no previous posts were made, the website should tell the user that no previous posts are available

- Before login, the user must not be able to see options to:
  - Add a post
  - Update their profile

ADDING A POST – MODEL IS EASY

- We already had the model ready (from week 10)

```
class Post(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    body = db.Column(db.String(140))
    timestamp = db.Column(db.DateTime, index=True, default=datetime.utcnow)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))

    def __repr__(self):
        return '<Post made on {}: {}>'.format(self.timestamp,self.body)
```

ADDING A POST – CREATE A FORM

```
class PostForm(FlaskForm):
    postbody = StringField('MicroPost', validators=[DataRequired()])
    submit = SubmitField('Add Post')
```

ADDING A POST – CREATE A ROUTE

```
@app.route('/post', methods=['GET', 'POST'])
def post():
    form = PostForm()
    if not session.get("USERNAME") is None:
        if form.validate_on_submit():
            body = form.postbody.data
            user_in_db = User.query.filter(User.username == session.get("USERNAME")).first()
            post = Post(body=body, author = user_in_db)
            db.session.add(post)
            db.session.commit()
            return redirect(url_for('post'))
        return render_template('post.html', title='User Posts', prev_posts=prev_posts, form=form)
    else:
        flash("User needs to either login or signup first")
        return redirect(url_for('login'))
```

RECALL – ADDITIONAL REQUIREMENT

```
@app.route('/post', methods=['GET', 'POST'])
def post():
    form = PostForm()
    if not session.get("USERNAME") is None:
        if form.validate_on_submit():
            body = form.postbody.data
            user_in_db = User.query.filter(User.username == session.get("USERNAME")).first()
            post = Post(body=body, author = user_in_db)
            db.session.add(post)
            db.session.commit()
            return redirect(url_for('post'))
        else:
            user_in_db = User.query.filter(User.username == session.get("USERNAME")).first()
            prev_posts = Post.query.filter(Post.user_id == user_in_db.id).all()
            print("Checking for user: {} with id: {}".format(user_in_db.username, user_in_db.id))
            return render_template('post.html', title='User Posts', prev_posts=prev_posts, form=form)
    else:
        flash("User needs to either login or signup first")
        return redirect(url_for('login'))
```

ADDING A POST – SHOW THE TEMPLATE

```
{% extends "base.html" %}
{% block content %}
    {% if prev_posts %}
        <ul>
        {% for post in prev_posts %}
            <li> {{ post }}</li>
        {% endfor %}
        </ul>
        <hr>
    {% else %}
        <p>No previous posts!</p>
        <hr>
    {% endif %}
    <form action="" method="post" novalidate>
        {{ form.hidden_tag() }}
        <p>
            {{ form.postbody.label }}<br>
            {{ form.postbody(size=244) }}

            {% for error in form.postbody.errors %}
            <span style="color: red;">[{{ error }}]</span>
            {% endfor %}
        </p>
        <p>{{ form.submit() }}</p>
    </form>
```

ANOTHER REQUIREMENT

- Before login, the user must not be able to see options to:
  - Add a post
  - Update their profile

- This means that the base template must change. We need a new template for logged-in users

BASE.HTML

```
<body>
    <div>
        Microblog:
        <a href="{{ url_for('index') }}">Home</a>
        <a href="{{ url_for('signup') }}">Signup</a>
        <a href="{{ url_for('login') }}">Login</a>
        <a href="{{ url_for('static', filename='plain.html') }}">Static Page</a>
    </div>
    <hr>
    {% with messages = get_flashed_messages() %}
    {% if messages %}
    <ul>
```

LOGGEDIN_BASE.HTML

```
<html>
    <head>
        <title>microblog</title>
        <link rel="stylesheet" href="{{ url_for('static', filename='style/mystyle.css') }}">
    </head>
    <body>
        <div>
            Microblog:
            <a href="{{ url_for('post') }}">Write a Post</a>
            <a href="{{ url_for('profile') }}">Edit Profile</a>
            <a href="{{ url_for('logout') }}">Logout</a>
        </div>
        <hr>
        {% with messages = get_flashed_messages() %}
        {% if messages %}
```

CHANGE BOTH PROFILE.HTML AND POST.HTML

profile.html

```
{% extends "loggedin_base.html" %}

{% block content %}
    <p> Let's complete your profile:</p>

    <form action="" method="post" enctype="multipart/form-data" novalidate>
        {{ form.hidden_tag() }}
        <p>
            {{ form.dob.label }}<br>
            {{ form.dob(size=32) }}
```

post.html

```
{% extends "loggedin_base.html" %}

{% block content %}
    {% if prev_posts %}
        <ul>
        {% for post in prev_posts %}
            <li> {{ post }}</li>
        {% endfor %}
        </ul>
        <hr>
    {% else %}
        <p>No previous posts!</p>
        <hr>
```

BUT THAT'S NOT ENOUGH!

- We need to change `routes.py`

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
```

```
if (check_password_hash(user_in_db.password_hash, form.password.data)):
    flash('Login success!')
    session["USERNAME"] = user_in_db.username
    return redirect(url_for('profile'))
```

NEW ROUTES.PY

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
```

```
if (check_password_hash(user_in_db.password_hash, form.password.data)):
    flash('Login success!')
    session["USERNAME"] = user_in_db.username
    return redirect(url_for('choice'))
```

THEN WE NEED TO CREATE A CHOICE.HTML

```
{% extends "loggedin_base.html" %}

{% block content %}
    <p>You are logged in as {{user.username}}</p>
    <p>By clicking the relevant link, please make a choice of:
        <ul>
            <li><a href="{{ url_for('post') }}">Write a micro post</a></li>
            <li><a href="{{ url_for('profile') }}">Edit your profile</a></li>
        </ul>
    </p>
{% endblock %}
```

ARE WE DONE?

- Almost
- We have not written code to update our profile yet
- Updating profile means:
  1. We need to retrieve the previous profile of the logged-in user
  2. We need to show it on the form [What if he/she has not created a profile?]
  3. We need to make changes to this profile
  4. We need to save the changes
- The good thing about using Flask-WTForms and Flask-SQlAlchemy is that we do not need to write separate code for 2 & 3
- We just need to take care of 1 & 4 [That means only `routes.py` needs change]

NEW profile ROUTE

```python
@app.route('/profile', methods=['GET', 'POST'])
def profile():
    form = ProfileForm()
    if not session.get("USERNAME") is None:
        if form.validate_on_submit():
            #...
            # now we add the object to the database
            user_in_db = User.query.filter(User.username == session.get("USERNAME")).first()
            profile = Profile(dob=form.dob.data, gender=form.gender.data, cv=cv_filename,user=user_in_db)
            db.session.add(profile)
            # remember to commit
            db.session.commit()
            return redirect(url_for('choice'))
        else:
            user_in_db = User.query.filter(User.username == session.get("USERNAME")).first()
            stored_profile = Profile.query.filter(Profile.user == user_in_db).first()
            if not stored_profile:
                return render_template('profile.html', title='Add your profile', form=form)
            else:
                form.dob.data = stored_profile.dob
                form.gender.data = stored_profile.gender
                return render_template('profile.html', title='Modify your profile', form=form)
```

IS THIS OKAY?

- Are we done?

UPDATE THE PROFILE, NOT ADD TO IT

- Again, using Flask-SQLAlchemy, there's no need for any update query
- Simply retrieve the object
- Modify it in python
- And commit it

UPDATED PROFILE FUNCTION

```
    # now we add the object to the database
    user_in_db = User.query.filter(User.username == session.get("USERNAME")).first()
    #check if user already has a profile
    stored_profile = Profile.query.filter(Profile.user == user_in_db).first()
    if not stored_profile:
        # if no profile exists, add a new object
        profile = Profile(dob=form.dob.data, gender=form.gender.data, cv=cv_filename,user=user
        db.session.add(profile)
    else:
        # else, modify the existing object with form data
        stored_profile.dob = form.dob.data
        stored_profile.gender = form.gender.data
    # remember to commit
    db.session.commit()
    return redirect(url_for('choice'))
else:
    user_in_db = User.query.filter(User.username == session.get("USERNAME")).first()
    stored_profile = Profile.query.filter(Profile.user == user_in_db).first()
    if not stored_profile:
        return render_template('profile.html', title='Add your profile', form=form)
    else:
        form.dob.data = stored_profile.dob
        form.gender.data = stored_profile.gender
        return render_template('profile.html', title='Modify your profile', form=form)
```

TO-DO IN CLASS

- Download the code
- Create a new database using the flask shell
- Add new data to the database and check if it works

- Next Lab (Thursday – 28th November): Flask Graded Assignment!