

# LECTURE 38

## WEB APP WRAP-UP

MCS 275 Spring 2021

Emily Dumas

# LECTURE 38: WEB APP WRAP-UP

Course bulletins:

- [Project 4 description](#) posted. Make a Flask+SQLite app. Very flexible rules (e.g. can give and receive help, use online resources, base it on Yellaro/Whinge or not).
- Project 4 is due 6pm CDT Friday April 30.

This is the last in our lecture series focused on live coding two web applications (source [here](#)):

- Yellaro - Simple chat application
- Whinge - Submit your pet peeves, vote on others' submissions

# CHAT APP TODO

- ✓ HTML mockup
- ✓ \* Stylesheet
- ✓ Database schema & test data
- ✓ Flask route to generate front page
- ✓ \* Add form to post a message to HTML
- ✓ Flask route for new message submission

\* Additional refinements outside lecture.

# VOTE APP TODO

- ✓ \* HTML mockup
- ✓ \* Stylesheet
- ✓ \* Database schema & test data
- ✓ \* Flask route to generate front page
- ✓ \* Form to submit new item
- ✓ \* Flask route to create new item
- Make the + and - buttons work
- Second view (chrono vs score)

\* Copied from chat app with minor changes.

# CHAT APP ROUTES

- / - (GET) show message feed
- /**post** - (POST) add message

# VOTE APP ROUTES

- `/top/` - (GET) show items, ranked
- `/new/` - (GET) show items, chrono
- `/post` - (POST) submit item
- `/plus?postid=15` - (GET) score += 1
- `/minus?postid=15` - (GET) score -= 1

# FLASK FUNCTIONS

- `url_for(func_name, param1=val1, param2=val2, ...)` - Get URL corresponding to a function within this application, with optional query parameters, e.g.  
`url_for("record_score", postid=5, score=11)` might return `"/setscore?postid=5&score=11"` if your app contains:

```
@app.route("/setscore")
def record_score():
    print("recording score {} for postid {}".format(
        flask.request.values.get("score"),
        flask.request.values.get("postid"),
    ))
```

# FLASK FUNCTIONS

- **`redirect(url)`** - *Returning this object from a route will cause the HTTP server to issue a 302 response, redirecting the client to `url`. (Basically, it means "ask them to load a different URL")*
- **`abort(http_error_code)`** - *Immediately stop and return a HTTP error code (usually 400 bad request, 401 not authorized, 403 forbidden, or 404 not found).*

# RETROSPECTIVE

Some of the things you'd do differently in a "real" application:

- **Templates:** Instead of strings containing HTML tags embedded in the Python application code, we'd write HTML *templates* that the Flask app would *render*.
- **Cookies:** login page checks credentials against DB, sets browser *cookie*. Auth-required pages check for it, redirect to login page if not found.
- **JavaScript:** e.g. to check for new messages in real time, post new message without loading a new page
- **Pagination:** Links to show next/prev page of messages or posts.
- **Better HTML+CSS:** Bottom of the message feed flush with bottom of div. Maybe infinite scrollbar with JS?
- **Better vote schema:** Instead of storing vote counts (or in addition to them), store a table of user-vote tuples. Allows vote policy enforcement (e.g. one vote per user).

# REFERENCES

- [jsfiddle](#) - Write and test HTML+CSS quickly in browser
- [HTML tutorial from w3schools](#)
- [CSS tutorial from w3schools](#)
- [The Flask tutorial](#)

# REVISION HISTORY

- 2021-04-16 Initial publication

