Building a Weather App with Flask

Clara Bennett & Emily Williamson

Step 1: Initialize app

```
git checkout 0408b8
```

Create minimal Flask app with route `/weather`

Step 2: Reorganize and add template

`git checkout f40033`

- Why reorganize?
  - Cleaner organization for adding future functionality
- Render templater
  - Add basic HTML form elements to our template
  - Add (non-functional) Submit button
  - Return `render_template` function instead of string

Step 3: Add functionality

```
git checkout 84f6f5
```

- Pass form data into our template on submit
  - Add `name='zip_code'` to our template form
  - Assign a variable in our route to input `zip_code` from our template form
  - Pass new variable to `render_template` function
  - Highlight error for variable definition in route
- What is `request`?
  - A global object within your Flask application that holds the information from the current HTTP request

Step 4: Add endpoint for submit

`git checkout 8ad2f0`

- Create a new route for "submit"
  - Redirect `POST` from `/weather` to new route
  - Render new template associated with redirect route
- Note that our main `weather_app.py` file is starting to accumulate some "clutter"[1]

[1] Not actually so bad, because the app is simple and small, but it's enough for illustrative purposes

Step 5: De-clutter

`git checkout 258bcd`

- Pull routes (a.k.a. "views" or "controllers") into a separate file
    - Use `Blueprint` object to register routes on our Flask app
    - Decouple logic from app configuration
    - Does not change functionality of the app
- Blueprint potential gotcha: route namespacing

Step 6: Add WTForms

```
git checkout 771d18
```

- Why WTForms?
  - Reusability
  - Field types, validation
  - Ease of passing information through app
- Implementation
  - Create `LocationForm` class and inherit from `Form`
  - Validators on `zip_code` template field, in `forms.py`
  - Use `LocationForm` in routes instead of `request.form` data
  - Iterate over form fields in template

Step 7: Add validation errors

```
git checkout e84a29
```

- Add custom error message to form validator
- Add error logic to template
- Add styling to error messages
    - In real life, styles should be separated from templates

Disclaimer: we are not expert front-end devs!

Step 8: Add API call

`git checkout e1b9bd`

- Create `get_weather` function in new file
  - Use `requests` library for easy HTTP call handling
  - Information from API is returned as JSON
- Update `show_weater` function
  - Pass `location` function to be used in API call
  - Use API response data in `render_template`
  - Add API response error page

Step 9: Your turn!

What to change?

* Grabe more information from API response and display on `location_weather.html` template

* Add to `LocationForm`

* STYLE IT UP!

* Whatever you want

Resources

- General Flask documentation
- Flask Blueprint docs
- WTForms docs
- Wunderground API Reference