Getting Stated with Flask Tutorial

In this tutorial we will create a Flask miniblogger application

1. Development Platform and Setup

Local windows environment (powershell), python 3 (idle editor), and Flask

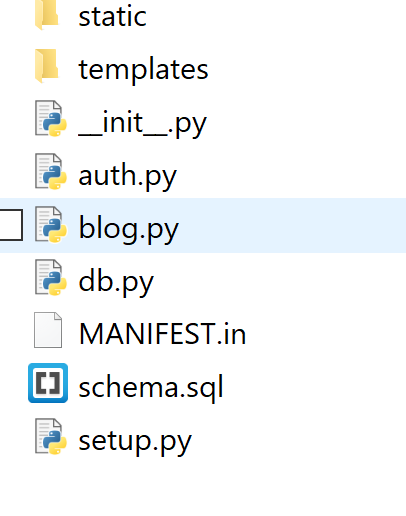
Commands are all done in powershell

check for python & pip version ($> python --version ...

install Flask: pip install Flask

2. Layout

File Structure within our workspace folder(flaskr)



static

|-> style.css

templates

|-> auth

|-> login.html

|-> register.html

|->blog

|-> create.html

|-> index.html

|-> update.html

|->base.html

3. git & github repositories

* Create a github account [here](https://github.com/join?source=experiment-header-dropdowns-home)
* Create a new repository set as public, no readme
  + Copy the command: $> git remote install origin fancy\_repo\_name

* On your local device navigate to the folder you wish to work in and use the following commands

git init

git add .

git commit -m "really special message"

git push origin master

Entire flask repo: https://github.com/pallets/flask

**4 Flask Application Factory**

* **Create the \_\_init\_\_.py in flaskr folder**

import os

from flask import Flask

def create\_app(test\_config=None):

# create and configure the app

app = Flask(\_\_name\_\_, instance\_relative\_config=True)

app.config.from\_mapping(

SECRET\_KEY='dev',

DATABASE=os.path.join(app.instance\_path, 'flaskr.sqlite'),

)

from . import db

db.init\_app(app)

from . import auth

app.register\_blueprint(auth.bp)

from . import blog

app.register\_blueprint(blog.bp)

app.add\_url\_rule('/', endpoint='index')

return app

if test\_config is None:

# load the instance config, if it exists, when not testing

app.config.from\_pyfile('config.py', silent=True)

else:

# load the test config if passed in

app.config.from\_mapping(test\_config)

# ensure the instance folder exists

try:

os.makedirs(app.instance\_path)

except OSError:

pass

# a simple page that says hello

@app.route('/hello')

def hello():

return 'Hello, World!'

return app

* **Running the 'factory' but only get errors as incomplete these commands are done in powershell**

app = Flask(\_\_init.py\_\_, instance\_relative\_config=True)

$env:Flask\_APP= "flaskr"

$env:Flask\_ENV= "development"

flask run

4. Application sqlite3 database

* **Create the db.py in flaskr FOLDER**

import sqlite3

import click

from flask import current\_app, g

from flask.cli import with\_appcontext

def init\_db():

db = get\_db()

with current\_app.open\_resource('schema.sql') as f:

db.executescript(f.read().decode('utf8'))

def init\_app(app):

app.teardown\_appcontext(close\_db)

app.cli.add\_command(init\_db\_command)

@click.command('init-db')

@with\_appcontext

def init\_db\_command():

"""Clear the existing data and create new tables."""

init\_db()

click.echo('Initialized the database.')

def get\_db():

if 'db' not in g:

g.db = sqlite3.connect(

current\_app.config['DATABASE'],

detect\_types=sqlite3.PARSE\_DECLTYPES

)

g.db.row\_factory = sqlite3.Row

return g.db

def close\_db(e=None):

db = g.pop('db', None)

if db is not None:

db.close()

* **Create schema.sql**

DROP TABLE IF EXISTS user;

DROP TABLE IF EXISTS post;

CREATE TABLE user (

id INTEGER PRIMARY KEY AUTOINCREMENT,

username TEXT UNIQUE NOT NULL,

password TEXT NOT NULL

);

CREATE TABLE post (

id INTEGER PRIMARY KEY AUTOINCREMENT,

author\_id INTEGER NOT NULL,

created TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

title TEXT NOT NULL,

body TEXT NOT NULL,

FOREIGN KEY (author\_id) REFERENCES user (id)

);

* Initialize the data base using: flask init-db

**5. Security program controller (auth.py)**

* **Create auth.py in flaskr folder**

import functools

from flask import (

Blueprint, flash, g, redirect, render\_template, request, session, url\_for

)

from werkzeug.security import check\_password\_hash, generate\_password\_hash

from flaskr.db import get\_db

bp = Blueprint('auth', \_\_name\_\_, url\_prefix='/auth')

@bp.route('/register', methods=('GET', 'POST'))

def register():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

db = get\_db()

error = None

if not username:

error = 'Username is required.'

elif not password:

error = 'Password is required.'

elif db.execute(

'SELECT id FROM user WHERE username = ?', (username,)

).fetchone() is not None:

error = 'User {} is already registered.'.format(username)

if error is None:

db.execute(

'INSERT INTO user (username, password) VALUES (?, ?)',

(username, generate\_password\_hash(password))

)

db.commit()

return redirect(url\_for('auth.login'))

flash(error)

return render\_template('auth/register.html')

@bp.route('/login', methods=('GET', 'POST'))

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

db = get\_db()

error = None

user = db.execute(

'SELECT \* FROM user WHERE username = ?', (username,)

).fetchone()

if user is None:

error = 'Incorrect username.'

elif not check\_password\_hash(user['password'], password):

error = 'Incorrect password.'

if error is None:

session.clear()

session['user\_id'] = user['id']

return redirect(url\_for('index'))

flash(error)

return render\_template('auth/login.html')

@bp.before\_app\_request

def load\_logged\_in\_user():

user\_id = session.get('user\_id')

if user\_id is None:

g.user = None

else:

g.user = get\_db().execute(

'SELECT \* FROM user WHERE id = ?', (user\_id,)

).fetchone()

@bp.route('/logout')

def logout():

session.clear()

return redirect(url\_for('index'))

def login\_required(view):

@functools.wraps(view)

def wrapped\_view(\*\*kwargs):

if g.user is None:

return redirect(url\_for('auth.login'))

return view(\*\*kwargs)

return wrapped\_view

6. Blog interface controller (blog.py)

* **Create blog.py in flaskr**

from flask import (

Blueprint, flash, g, redirect, render\_template, request, url\_for

)

from werkzeug.exceptions import abort

from flaskr.auth import login\_required

from flaskr.db import get\_db

bp = Blueprint('blog', \_\_name\_\_)

@bp.route('/create', methods=('GET', 'POST'))

@login\_required

def create():

if request.method == 'POST':

title = request.form['title']

body = request.form['body']

error = None

if not title:

error = 'Title is required.'

if error is not None:

flash(error)

else:

db = get\_db()

db.execute(

'INSERT INTO post (title, body, author\_id)'

' VALUES (?, ?, ?)',

(title, body, g.user['id'])

)

db.commit()

return redirect(url\_for('blog.index'))

@bp.route('/<int:id>/update', methods=('GET', 'POST'))

@login\_required

def update(id):

post = get\_post(id)

if request.method == 'POST':

title = request.form['title']

body = request.form['body']

error = None

if not title:

error = 'Title is required.'

if error is not None:

flash(error)

else:

db = get\_db()

db.execute(

'UPDATE post SET title = ?, body = ?'

' WHERE id = ?',

(title, body, id)

)

db.commit()

return redirect(url\_for('blog.index'))

return render\_template('blog/update.html', post=post)

@bp.route('/<int:id>/delete', methods=('POST',))

@login\_required

def delete(id):

get\_post(id)

db = get\_db()

db.execute('DELETE FROM post WHERE id = ?', (id,))

db.commit()

return redirect(url\_for('blog.index'))

7. Views and Templates

* Create the static sub-folder
  + Inside create style.css

html { font-family: sans-serif; background: #eee; padding: 1rem; }

body { max-width: 960px; margin: 0 auto; background: white; }

h1 { font-family: serif; color: #377ba8; margin: 1rem 0; }

a { color: #377ba8; }

hr { border: none; border-top: 1px solid lightgray; }

nav { background: lightgray; display: flex; align-items: center; padding: 0 0.5rem; }

nav h1 { flex: auto; margin: 0; }

nav h1 a { text-decoration: none; padding: 0.25rem 0.5rem; }

nav ul { display: flex; list-style: none; margin: 0; padding: 0; }

nav ul li a, nav ul li span, header .action { display: block; padding: 0.5rem; }

.content { padding: 0 1rem 1rem; }

.content > header { border-bottom: 1px solid lightgray; display: flex; align-items: flex-end; }

.content > header h1 { flex: auto; margin: 1rem 0 0.25rem 0; }

.flash { margin: 1em 0; padding: 1em; background: #cae6f6; border: 1px solid #377ba8; }

.post > header { display: flex; align-items: flex-end; font-size: 0.85em; }

.post > header > div:first-of-type { flex: auto; }

.post > header h1 { font-size: 1.5em; margin-bottom: 0; }

.post .about { color: slategray; font-style: italic; }

.post .body { white-space: pre-line; }

.content:last-child { margin-bottom: 0; }

.content form { margin: 1em 0; display: flex; flex-direction: column; }

.content label { font-weight: bold; margin-bottom: 0.5em; }

.content input, .content textarea { margin-bottom: 1em; }

.content textarea { min-height: 12em; resize: vertical; }

input.danger { color: #cc2f2e; }

input[type=submit] { align-self: start; min-width: 10em; }

* Create the templates folder
  + Add base.html to this folder

<!doctype html>

<title>{% block title %}{% endblock %} - Flaskr</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

<nav>

<h1>Flaskr</h1>

<ul>

{% if g.user %}

<li><span>{{ g.user['username'] }}</span>

<li><a href="{{ url\_for('auth.logout') }}">Log Out</a>

{% else %}

<li><a href="{{ url\_for('auth.register') }}">Register</a>

<li><a href="{{ url\_for('auth.login') }}">Log In</a>

{% endif %}

</ul>

</nav>

<section class="content">

<header>

{% block header %}{% endblock %}

</header>

{% for message in get\_flashed\_messages() %}

<div class="flash">{{ message }}</div>

{% endfor %}

{% block content %}{% endblock %}

</section>

* + Add sub folder auth to templates
    - Add login.html to auth folder

{% extends 'base.html' %}

{% block header %}

<h1>{% block title %}Log In{% endblock %}</h1>

{% endblock %}

{% block content %}

<form method="post">

<label for="username">Username</label>

<input name="username" id="username" required>

<label for="password">Password</label>

<input type="password" name="password" id="password" required>

<input type="submit" value="Log In">

</form>

{% endblock %}

* + - Add register.html to auth folder

{% extends 'base.html' %}

{% block header %}

<h1>{% block title %}Register{% endblock %}</h1>

{% endblock %}

{% block content %}

<form method="post">

<label for="username">Username</label>

<input name="username" id="username" required>

<label for="password">Password</label>

<input type="password" name="password" id="password" required>

<input type="submit" value="Register">

</form>

{% endblock %}

* + Add sub folder blog to templates
    - Add create.html to blog folder

{% extends 'base.html' %}

{% block header %}

<h1>{% block title %}New Post{% endblock %}</h1>

{% endblock %}

{% block content %}

<form method="post">

<label for="title">Title</label>

<input name="title" id="title" value="{{ request.form['title'] }}" required>

<label for="body">Body</label>

<textarea name="body" id="body">{{ request.form['body'] }}</textarea>

<input type="submit" value="Save">

</form>

{% endblock %}

* + - Add index.html to blog folder

{% extends 'base.html' %}

{% block header %}

<h1>{% block title %}Posts{% endblock %}</h1>

{% if g.user %}

<a class="action" href="{{ url\_for('blog.create') }}">New</a>

{% endif %}

{% endblock %}

{% block content %}

{% for post in posts %}

<article class="post">

<header>

<div>

<h1>{{ post['title'] }}</h1>

<div class="about">by {{ post['username'] }} on {{ post['created'].strftime('%Y-%m-%d') }}</div>

</div>

{% if g.user['id'] == post['author\_id'] %}

<a class="action" href="{{ url\_for('blog.update', id=post['id']) }}">Edit</a>

{% endif %}

</header>

<p class="body">{{ post['body'] }}</p>

</article>

{% if not loop.last %}

<hr>

{% endif %}

{% endfor %}

{% endblock %}

* + - Add update.html to blog folder

{% extends 'base.html' %}

{% block header %}

<h1>{% block title %}Edit "{{ post['title'] }}"{% endblock %}</h1>

{% endblock %}

{% block content %}

<form method="post">

<label for="title">Title</label>

<input name="title" id="title"

value="{{ request.form['title'] or post['title'] }}" required>

<label for="body">Body</label>

<textarea name="body" id="body">{{ request.form['body'] or post['body'] }}</textarea>

<input type="submit" value="Save">

</form>

<hr>

<form action="{{ url\_for('blog.delete', id=post['id']) }}" method="post">

<input class="danger" type="submit" value="Delete" onclick="return confirm('Are you sure?');">

</form>

{% endblock %}

8. Running flaskr

In powershell, navigate to one folder above flaskr. From there run the following commands

app = Flask(\_\_init.py\_\_, instance\_relative\_config=True)

$env:Flask\_APP= "flaskr"

$env:Flask\_ENV= "development"

Flask run

* In your web browser, navigate to localhost.
* Test all routes created
* You can stop the server using Ctrl+C

9. Remember to save all code in repository!

Link to example Repo: <https://github.com/jtonto/classGit>

If you run into issues, try cloning this repo