PennyAdmin08aHash/penny.sql (Page 1 of 1)

```
 1: DROP TABLE IF EXISTS books;
 2: CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);
 3: INSERT INTO books (isbn, author, title)
 4:    VALUES ('123', 'Kernighan','The Practice of Programming');
 5: INSERT INTO books (isbn, author, title)
 6:    VALUES ('234', 'Kernighan','The C Programming Language');
 7: INSERT INTO books (isbn, author, title)
 8:    VALUES ('345', 'Sedgewick','Algorithms in C');
 9:
10: DROP TABLE IF EXISTS users;
11: CREATE TABLE users (username TEXT PRIMARY KEY, password TEXT);
12: INSERT INTO users (username, password) VALUES ('rdondero',
13:    'cd2eb0837c9b4c962c22d2ff8b5441b7b45805887f051d39bf133b583baf6860');
14: INSERT INTO users (username, password) VALUES ('bwk',
15:    'f2afd1cacb5441a5e65a7a460a5f9898b7b98b08aa6323a2e53c8b9a9686cd86');
16: INSERT INTO users (username, password) VALUES ('rs',
17:    '17f165d5a5ba695f27c023a83aa2b3463e23810e360b7517127e90161eebabda');
18:
19: DROP TABLE IF EXISTS authorizedusers;
20: CREATE TABLE authorizedusers (username TEXT PRIMARY KEY);
21: INSERT INTO authorizedusers (username) VALUES ('rdondero');
22: INSERT INTO authorizedusers (username) VALUES ('bwk');
```

blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

PennyAdmin08aHash/auth.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# auth.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import hashlib
import flask
import database

from top import app

#-----------------------------------------------------------------------

def _hash(password):

    hash_code = hashlib.sha256()
    hash_code.update(password.encode('ascii'))
    hash_code = hash_code.hexdigest()
    return hash_code

#-----------------------------------------------------------------------

def _valid_username_and_password(username, password):

    stored_hash_code = database.get_password(username)
    if stored_hash_code is None:
        return False
    hash_code = _hash(password)
    return hash_code == stored_hash_code

#-----------------------------------------------------------------------

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

    msg = flask.request.args.get('msg')
    if msg is None:
        msg = ''

    html = flask.render_template('login.html', msg=msg)

    response = flask.make_response(html)
    return response

#-----------------------------------------------------------------------

@app.route('/handlelogin', methods=['POST'])
def handle_login():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if not _valid_username_and_password(username, password):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    original_url = flask.session.get('original_url', '/index')
    response = flask.redirect(original_url)
    flask.session['username'] = username
```

PennyAdmin08aHash/auth.py (Page 2 of 2)

```
    return response

#-----------------------------------------------------------------------

@app.route('/logout', methods=['GET'])
def logout():

    flask.session.clear()
    html_code = flask.render_template('loggedout.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/signup', methods=['GET'])
def signup():

    error_msg = flask.request.args.get('error_msg')
    if error_msg is None:
        error_msg = ''

    html_code = flask.render_template('signup.html',
        error_msg=error_msg)

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handlesignup', methods=['POST'])
def handle_signup():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid username'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid password'))

    hash_code = _hash(password)
    successful = database.add_user(username, hash_code)
    if not successful:
        return flask.redirect(
            flask.url_for('signup', error_msg='Duplicate username'))

    return flask.redirect(
        flask.url_for('login', msg='You now are signed up.'))

#-----------------------------------------------------------------------

def get_username():

    return flask.session.get('username')

#-----------------------------------------------------------------------

def authenticate():

    username = flask.session.get('username')
    if username is None:
        response = flask.redirect(flask.url_for('login'))
        flask.session['original_url'] = flask.request.url
        flask.abort(response)
```

PennyAdmin08bSaltHash/penny.sql (Page 1 of 1)

```
    1: DROP TABLE IF EXISTS books;
    2: CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);
    3: INSERT INTO books (isbn, author, title)
    4:    VALUES ('123', 'Kernighan','The Practice of Programming');
    5: INSERT INTO books (isbn, author, title)
    6:    VALUES ('234', 'Kernighan','The C Programming Language');
    7: INSERT INTO books (isbn, author, title)
    8:    VALUES ('345', 'Sedgewick','Algorithms in C');
    9:
   10: DROP TABLE IF EXISTS users;
   11: CREATE TABLE users (username TEXT PRIMARY KEY, password TEXT);
   12: INSERT INTO users (username, password) VALUES ('rdondero',
   13:    'pbkdf2:sha256:600000$UOVCfeB4bjAW4nYx$0641776b12a4054fe5fb72eb4f9bbf
73c4266099de5ec01f09c325ed56746c3a');
   14: INSERT INTO users (username, password) VALUES ('bwk',
   15:    'pbkdf2:sha256:600000$fQwzwUw8ZCPET43r$2390d394f64f239e5bc765f4163d49
d40b97601fae4d6ae0830c52296438e6ae');
   16: INSERT INTO users (username, password) VALUES ('rs',
   17:    'pbkdf2:sha256:600000$xbvn5wij1R8eJ5Mq$5bb39b39b45d65628091f4ba5fabda
99b30e9a60c447818db626f35f1dffdf9f');
   18:
   19: DROP TABLE IF EXISTS authorizedusers;
   20: CREATE TABLE authorizedusers (username TEXT PRIMARY KEY);
   21: INSERT INTO authorizedusers (username) VALUES ('rdondero');
   22: INSERT INTO authorizedusers (username) VALUES ('bwk');
```

blank (Page 1 of 1)

```
    1: This page is intentionally blank.
```

PennyAdmin08bSaltHash/auth.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# auth.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import werkzeug.security
import flask
import database

from top import app

#-----------------------------------------------------------------------

def _valid_username_and_password(username, password):

    stored_hash_code = database.get_password(username)
    if stored_hash_code is None:
        return False
    return werkzeug.security.check_password_hash(
        stored_hash_code, password)

#-----------------------------------------------------------------------

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

    msg = flask.request.args.get('msg')
    if msg is None:
        msg = ''

    html = flask.render_template('login.html', msg=msg)

    response = flask.make_response(html)
    return response

#-----------------------------------------------------------------------

@app.route('/handlelogin', methods=['POST'])
def handle_login():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    if not _valid_username_and_password(username, password):
        return flask.redirect(
            flask.url_for('login', msg='Wrong username or password'))
    original_url = flask.session.get('original_url', '/index')
    response = flask.redirect(original_url)
    flask.session['username'] = username
    return response

#-----------------------------------------------------------------------

@app.route('/logout', methods=['GET'])
def logout():

    flask.session.clear()
    html_code = flask.render_template('loggedout.html')
```

PennyAdmin08bSaltHash/auth.py (Page 2 of 2)

```
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/signup', methods=['GET'])
def signup():

    error_msg = flask.request.args.get('error_msg')
    if error_msg is None:
        error_msg = ''

    html_code = flask.render_template('signup.html',
        error_msg=error_msg)

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handlesignup', methods=['POST'])
def handle_signup():

    username = flask.request.form.get('username')
    password = flask.request.form.get('password')
    if (username is None) or (username.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid username'))
    if (password is None) or (password.strip() == ''):
        return flask.redirect(
            flask.url_for('signup', error_msg='Invalid password'))

    hash_code = werkzeug.security.generate_password_hash(
        password, 'pbkdf2')
    successful = database.add_user(username, hash_code)
    if not successful:
        return flask.redirect(
            flask.url_for('signup', error_msg='Duplicate username'))

    return flask.redirect(
        flask.url_for('login', msg='You now are signed up.'))

#-----------------------------------------------------------------------

def get_username():

    return flask.session.get('username')

#-----------------------------------------------------------------------

def authenticate():

    username = flask.session.get('username')
    if username is None:
        response = flask.redirect(flask.url_for('login'))
        flask.session['original_url'] = flask.request.url
        flask.abort(response)
```

PennyAdmin09aHttps/penny.py (Page 1 of 3)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import dotenv
import flask
import flask_session
import flask_sqlalchemy
import flask_wtf.csrf
import database
import auth
from top import app

#-----------------------------------------------------------------------

dotenv.load_dotenv()
app.secret_key = os.environ['APP_SECRET_KEY']
flask_wtf.csrf.CSRFProtect(app)

# Session configuration to store session data in the file system
# (simple, but not realistic for a deployed app):

#app.config['SESSION_PERMANENT'] = False
#app.config['SESSION_TYPE'] = 'filesystem'
#flask_session.Session(app)

# Session configuration to store session data in a database:

session_database_url = os.getenv('SESSION_DATABASE_URL',
    'sqlite:///pennysessions.sqlite')
session_database_url = session_database_url.replace(
    'postgres://', 'postgresql://')
app.config['SESSION_PERMANENT'] = False
app.config['SESSION_TYPE'] = 'sqlalchemy'
app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
flask_session.Session(app)

#-----------------------------------------------------------------------

@app.before_request
def before_request():
    is_running_locally = '//localhost:' in flask.request.url_root
    is_using_https = flask.request.is_secure
    if (not is_running_locally) and (not is_using_https):
        url = flask.request.url.replace('http://', 'https://', 1)
        return flask.redirect(url, code=301)
    return None

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------
```

PennyAdmin09aHttps/penny.py (Page 2 of 3)

```

@app.route('/menu', methods=['GET'])
def menu():

    auth.authenticate()
    username = auth.get_username()

    is_authorized = database.is_authorized(username)

    html_code = flask.render_template('menu.html', username=username,
        is_authorized=is_authorized)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/show', methods=['GET'])
def show():

    auth.authenticate()
    username = auth.get_username()

    books = database.get_books()
    html_code = flask.render_template('show.html',
        username=username, books=books)
    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/add', methods=['GET'])
def add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    html_code = flask.render_template('add.html', username=username)

    response = flask.make_response(html_code)
    return response

#-----------------------------------------------------------------------

@app.route('/handleadd', methods=['POST'])
def handle_add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        message = 'The addition was unsuccessful: missing ISBN'
    else:
        author = flask.request.form.get('author')
        if (author is None) or (author.strip() == ''):
```

PennyAdmin09aHttps/penny.py (Page 3 of 3)

```
131:             message = 'The addition was unsuccessful: missing author'
132:         else:
133:             title = flask.request.form.get('title')
134:             if (title is None) or (title.strip() == ''):
135:                 message = 'The addition was unsuccessful: missing title'
136:             else:
137:                 isbn = isbn.strip()
138:                 author = author.strip()
139:                 title = title.strip()
140:
141:                 successful = database.add_book(isbn, author, title)
142:                 if not successful:
143:                     message = 'The addition was unsuccessful: '
144:                     message += 'duplicate ISBN'
145:                 else:
146:                     message = 'The addition was successful'
147:
148:     html_code = flask.render_template('reportresults.html',
149:         username=username, message=message)
150:     response = flask.make_response(html_code)
151:     return response
152:
153: #-----------------------------------------------------------------------
154:
155: @app.route('/delete', methods=['GET'])
156: def delete():
157:
158:     auth.authenticate()
159:     username = auth.get_username()
160:
161:     if not database.is_authorized(username):
162:         html_code = 'You are not authorized to delete books.'
163:         response = flask.make_response(html_code)
164:         return response
165:
166:     html_code = flask.render_template('delete.html', username=username)
167:
168:     response = flask.make_response(html_code)
169:     return response
170:
171: #-----------------------------------------------------------------------
172:
173: @app.route('/handledelete', methods=['POST'])
174: def handle_delete():
175:
176:     auth.authenticate()
177:     username = auth.get_username()
178:
179:     if not database.is_authorized(username):
180:         html_code = 'You are not authorized to delete books.'
181:         response = flask.make_response(html_code)
182:         return response
183:
184:     isbn = flask.request.form.get('isbn')
185:     if (isbn is None) or (isbn.strip() == ''):
186:         message = 'The deletion was unsuccessful: missing ISBN'
187:     else:
188:         isbn = isbn.strip()
189:         database.delete_book(isbn)
190:         message = 'The deletion was successful'
191:
192:     html_code = flask.render_template('reportresults.html',
193:         username=username, message=message)
194:     response = flask.make_response(html_code)
195:     return response
```

blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

PennyAdmin09bHttps/penny.py (Page 1 of 3)

```
#!/usr/bin/env python

#------------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#------------------------------------------------------------------------

import os
import dotenv
import flask
import flask_session
import flask_sqlalchemy
import flask_wtf.csrf
import flask_talisman
import database
import auth

from top import app

#------------------------------------------------------------------------

dotenv.load_dotenv()
app.secret_key = os.environ['APP_SECRET_KEY']
flask_wtf.csrf.CSRFProtect(app)
flask_talisman.Talisman(app)

# Session configuration to store session data in the file system
# (simple, but not realistic for a deployed app):

#app.config['SESSION_PERMANENT'] = False
#app.config['SESSION_TYPE'] = 'filesystem'
#flask_session.Session(app)

# Session configuration to store session data in a database:

session_database_url = os.getenv('SESSION_DATABASE_URL',
    'sqlite:///pennysessions.sqlite')
session_database_url = session_database_url.replace(
    'postgres://', 'postgresql://')
app.config['SESSION_PERMANENT'] = False
app.config['SESSION_TYPE'] = 'sqlalchemy'
app.config['SQLALCHEMY_DATABASE_URI'] = session_database_url
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['SESSION_SQLALCHEMY'] = flask_sqlalchemy.SQLAlchemy(app)
flask_session.Session(app)

#------------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    html_code = flask.render_template('index.html')
    response = flask.make_response(html_code)
    return response

#------------------------------------------------------------------------

@app.route('/menu', methods=['GET'])
def menu():

    auth.authenticate()
    username = auth.get_username()

    is_authorized = database.is_authorized(username)
```

PennyAdmin09bHttps/penny.py (Page 2 of 3)

```

    html_code = flask.render_template('menu.html', username=username,
        is_authorized=is_authorized)
    response = flask.make_response(html_code)
    return response

#------------------------------------------------------------------------

@app.route('/show', methods=['GET'])
def show():

    auth.authenticate()
    username = auth.get_username()

    books = database.get_books()
    html_code = flask.render_template('show.html',
        username=username, books=books)
    response = flask.make_response(html_code)
    return response

#------------------------------------------------------------------------

@app.route('/add', methods=['GET'])
def add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    html_code = flask.render_template('add.html', username=username)

    response = flask.make_response(html_code)
    return response

#------------------------------------------------------------------------

@app.route('/handleadd', methods=['POST'])
def handle_add():

    auth.authenticate()
    username = auth.get_username()

    if not database.is_authorized(username):
        html_code = 'You are not authorized to add books.'
        response = flask.make_response(html_code)
        return response

    isbn = flask.request.form.get('isbn')
    if (isbn is None) or (isbn.strip() == ''):
        message = 'The addition was unsuccessful: missing ISBN'
    else:
        author = flask.request.form.get('author')
        if (author is None) or (author.strip() == ''):
            message = 'The addition was unsuccessful: missing author'
        else:
            title = flask.request.form.get('title')
            if (title is None) or (title.strip() == ''):
                message = 'The addition was unsuccessful: missing title'
            else:
                isbn = isbn.strip()
                author = author.strip()
```

PennyAdmin09bHttps/penny.py (Page 3 of 3)

```
131:                 title = title.strip()
132:
133:                 successful = database.add_book(isbn, author, title)
134:                 if not successful:
135:                     message = 'The addition was unsuccessful: '
136:                     message += 'duplicate ISBN'
137:                 else:
138:                     message = 'The addition was successful'
139:
140:     html_code = flask.render_template('reportresults.html',
141:         username=username, message=message)
142:     response = flask.make_response(html_code)
143:     return response
144:
145: #-----------------------------------------------------------------------
146:
147: @app.route('/delete', methods=['GET'])
148: def delete():
149:
150:     auth.authenticate()
151:     username = auth.get_username()
152:
153:     if not database.is_authorized(username):
154:         html_code = 'You are not authorized to delete books.'
155:         response = flask.make_response(html_code)
156:         return response
157:
158:     html_code = flask.render_template('delete.html', username=username)
159:
160:     response = flask.make_response(html_code)
161:     return response
162:
163: #-----------------------------------------------------------------------
164:
165: @app.route('/handledelete', methods=['POST'])
166: def handle_delete():
167:
168:     auth.authenticate()
169:     username = auth.get_username()
170:
171:     if not database.is_authorized(username):
172:         html_code = 'You are not authorized to delete books.'
173:         response = flask.make_response(html_code)
174:         return response
175:
176:     isbn = flask.request.form.get('isbn')
177:     if (isbn is None) or (isbn.strip() == ''):
178:         message = 'The deletion was unsuccessful: missing ISBN'
179:     else:
180:         isbn = isbn.strip()
181:         database.delete_book(isbn)
182:         message = 'The deletion was successful'
183:
184:     html_code = flask.render_template('reportresults.html',
185:         username=username, message=message)
186:     response = flask.make_response(html_code)
187:     return response
```

blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

PennyAdmin09cHttpsLocal/runserver.py (Page 1 of 1)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# runserver.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import penny

def main():

    if len(sys.argv) != 2:
        print('Usage: ' + sys.argv[0] + ' port', file=sys.stderr)
        sys.exit(1)

    try:
        port = int(sys.argv[1])
    except Exception:
        print('Port must be an integer.', file=sys.stderr)
        sys.exit(1)

    try:
        penny.app.run(host='0.0.0.0', port=port, debug=True,
            ssl_context=('cert.pem', 'key.pem'))
    except Exception as ex:
        print(ex, file=sys.stderr)
        sys.exit(1)

if __name__ == '__main__':
    main()
```

blank (Page 1 of 1)

```
This page is intentionally blank.
```