|  |
| --- |
| import os |
|  | import sqlite3 |
|  | from flask import Flask, request, session, g, redirect, url\_for, \ |
|  | abort, render\_template, flash |
|  | from datetime import datetime |
|  |  |
|  | # конфигурация |
|  | # DATABASE = 'data\_.db' |
|  | # DEBUG = True |
|  | # SECRET\_KEY = 'development key' |
|  | # USERNAME = 'admin' |
|  | # PASSWORD = 'default' |
|  |  |
|  | # далее, мы должны создать наше текущее приложение и инициализировать |
|  | # его в помощью конфигурации из того же файла, т. е. flaskr.py: |
|  | # создаём наше маленькое приложение :) |
|  |  |
|  | app = Flask(\_\_name\_\_) |
|  | # wsgi\_app = app.wsgi\_app |
|  | app.config.from\_object(\_\_name\_\_) |
|  |  |
|  | # Загружаем конфиг по умолчанию и переопределяем в конфигурации часть |
|  | # значений через переменную окружения |
|  | app.config.update(dict( |
|  | DATABASE=os.path.join(app.root\_path, 'data\_.db'), |
|  | DEBUG=True, |
|  | SECRET\_KEY='development key' |
|  |  |
|  | )) |
|  | app.config.from\_envvar('FLASKR\_SETTINGS', silent=True) |
|  |  |
|  | # Объект Config работает подобно словарю, поэтому мы можем обновлять его с помощью новых значений. |
|  |  |
|  | """обавим также метод, который позволяет простым способом соединиться с указанной базой данных. |
|  | Он может быть использован для открытия соединения по запросу, а также из интерактивной командной |
|  | оболочки Python или из скрипта. Это пригодится в дальнейшем. Мы создаём простое соединение |
|  | с базой данных SQLite и далее просим его использовать для представления строк объект sqlite3.Row. |
|  | Это позволит нам рассматривать строки, как если бы они были словарями, а не кортежами.""" |
|  |  |
|  |  |
|  | def connect\_db(): |
|  | print("connect\_db") |
|  | print(app.config['DATABASE']) |
|  | rv = sqlite3.connect(app.config['DATABASE']) |
|  | rv.row\_factory = sqlite3.Row |
|  | return rv |
|  |  |
|  |  |
|  | """функцию с именем init\_db, которая инициализирует базу данных. Позвольте, я сперва покажу вам код. |
|  | Просто добавьте в flaskr.py эту функцию после функции connect\_db:""" |
|  |  |
|  |  |
|  | def init\_db(db): |
|  | with app.open\_resource('shema.sql', mode='r') as f: |
|  | db.cursor().executescript(f.read()) |
|  | db.commit() |
|  | print('Initialized the database.') |
|  |  |
|  | """ Метод open\_resource() объекта приложения является удобной функцией-помощником, |
|  | которая откроет ресурс, обеспечиваемый приложением. Эта функция открывает файл из |
|  | места расположения ресурсов (в нашем случае папка flaskr\_n), и позволяет вам из него читать """ |
|  |  |
|  |  |
|  | # Всё, что вам надо знать на этот момент - это то, |
|  | # что вы можете безопасно сохранять информацию в объекте g. |
|  |  |
|  |  |
|  | def get\_db(): |
|  | """Если ещё нет соединения с базой данных, открыть новое - для |
|  | текущего контекста приложения""" |
|  | if not hasattr(g, 'sqlite\_db'): |
|  | g.sqlite\_db = connect\_db() |
|  | init\_db(g.sqlite\_db) |
|  | return g.sqlite\_db |
|  |  |
|  |  |
|  | """Flask обеспечил нас декоратором teardown\_appcontext(). Он выполняется каждый раз, |
|  | когда происходит разрыв контекста приложения:""" |
|  |  |
|  |  |
|  | @app.teardown\_appcontext |
|  | def close\_db(error): |
|  | """Функция, обозначенная как teardown\_appcontext() вызывается каждый раз при разрыве контекста приложения.""" |
|  | if hasattr(g, 'sqlite\_db'): |
|  | g.sqlite\_db.close() |
|  |  |
|  |  |
|  | """Функция представления передаёт записи в виде словаря шаблону list\_product.html |
|  | и возвращает сформированное отображение:""" |
|  |  |
|  |  |
|  | @app.route('/') |
|  | def list\_product(): |
|  | db = get\_db() |
|  | cur = db.execute( |
|  | 'select \* from products ' |
|  | 'order by id desc') |
|  | products = cur.fetchall() |
|  |  |
|  | if not session.get('logged\_in'): |
|  | return render\_template('list\_product.html', products=products, basket=[]) |
|  | cur = db.execute( |
|  | 'select ' |
|  | 'p.name as name, ' |
|  | 'p.model as model, ' |
|  | 'b.quantity as quantity, ' |
|  | 'b.id as basket\_Id, ' |
|  | 'p.price as price, ' |
|  | '(p.price \* b.quantity) as cost ' |
|  | 'from basket b, products p ' |
|  | 'where b.productsId = p.id ' |
|  | 'and b.userId=?', |
|  | [session['userId']]) |
|  | basket = cur.fetchall() |
|  |  |
|  | for row in basket: |
|  | for x in range(len(row)): |
|  | print(row[x]) |
|  |  |
|  | return render\_template('list\_product.html', products=products, basket=basket) |
|  |  |
|  |  |
|  | """Это представление позволяет пользователю, если он осуществил вход, добавлять |
|  | новые записи. Оно реагирует только на запросы типа POST, а фактическая форма |
|  | отображается на странице list\_product.""" |
|  |  |
|  |  |
|  | @app.route('/add\_product', methods=['POST']) |
|  | def add\_product(): |
|  | if not session.get('logged\_in'): |
|  | abort(401) |
|  | db = get\_db() |
|  | db.execute( |
|  | 'insert into products(name, model, price, quantity)' |
|  | ' values (?, ?, ?, ?)', |
|  | [request.form['name'], request.form['model'], |
|  | request.form['price'], request.form['quantity']]) |
|  |  |
|  | db.commit() |
|  | flash('New entry was successfully posted') |
|  | return redirect(url\_for('list\_product')) |
|  |  |
|  |  |
|  | @app.route('/delete-product', methods=['POST']) |
|  | def delete\_product(): |
|  | if not session.get('logged\_in'): |
|  | abort(401) |
|  | db = get\_db() |
|  | db.execute( |
|  | 'delete from products ' |
|  | 'where id=?', [request.form['id']]) |
|  | db.commit() |
|  | return redirect(url\_for('list\_product')) |
|  |  |
|  |  |
|  | @app.route('/add\_basket', methods=['POST']) |
|  | def add\_basket(): |
|  | if not session.get('logged\_in'): |
|  | return render\_template('list\_product.html', products=[], basket=[]) |
|  | db = get\_db() |
|  | cur = db.execute( |
|  | 'select quantity ' |
|  | 'from products where id = ?', |
|  | [request.form['id']]) |
|  | quantityInStock = int(cur.fetchone()[0]) |
|  | print('quantiytInStock = ' + str(quantityInStock)) |
|  | quantityToBuy = int(request.form['quantity']) |
|  | print('quantityToBuy = ' + str(quantityToBuy)) |
|  |  |
|  | if quantityInStock < quantityToBuy: |
|  | flash('There are not enough goods stock!') |
|  | return redirect(url\_for('list\_product')) |
|  |  |
|  | quantityInStock = quantityInStock - quantityToBuy |
|  |  |
|  | print('quantityInStock = ' + str(quantityInStock)) |
|  |  |
|  | db.execute( |
|  | 'update products ' |
|  | 'set quantity = ? ' |
|  | 'where id = ?', |
|  | [quantityInStock, request.form['id']]) |
|  |  |
|  | existingProduct = db.execute( |
|  | 'select id ' |
|  | 'from basket ' |
|  | 'where userId = ? and productsId = ?', |
|  | [session['userId'], request.form['id']]) |
|  |  |
|  | bask = existingProduct.fetchone() |
|  |  |
|  | if bask: |
|  | db.execute( |
|  | 'update basket ' |
|  | 'set quantity = quantity + ? ' |
|  | 'where id = ? ', [quantityToBuy, bask['id']]) |
|  | else: |
|  | db.execute( |
|  | 'insert into basket (' |
|  | 'productsId, quantity, userId) ' |
|  | 'values (?, ?, ?)', |
|  | [request.form['id'], request.form['quantity'], |
|  | session['userId']]) |
|  |  |
|  | db.commit() |
|  | return redirect(url\_for('list\_product')) |
|  |  |
|  |  |
|  | @app.route('/delete\_basket', methods=['POST']) |
|  | def delete\_basket(): |
|  | if not session.get('logged\_in'): |
|  | return render\_template('list\_product.html', basket=[], products=[]) |
|  | db = get\_db() |
|  | cur = db.execute( |
|  | 'select productsId, quantity ' |
|  | 'from basket where id=?', |
|  | [request.form['id']]) |
|  | row = cur.fetchone() |
|  | productsId = int(row[0]) |
|  | returnQty = int(row[1]) |
|  |  |
|  | db.execute( |
|  | 'delete from basket ' |
|  | 'where id = ?', [request.form['id']]) |
|  | db.execute( |
|  | 'update products set quantity = quantity + ? ' |
|  | 'where id = ?', |
|  | [returnQty, productsId]) |
|  |  |
|  | db.commit() |
|  | return redirect(url\_for('list\_product')) |
|  |  |
|  |  |
|  | @app.route('/del\_bask', methods=['GET', 'POST']) |
|  | def del\_bask(): |
|  | if not session.get('logged\_in'): |
|  | return render\_template('list\_product.html.html', basket=[], products=[]) |
|  | db = get\_db() |
|  |  |
|  | history = [] |
|  |  |
|  | if request.form: |
|  |  |
|  | historCard = db.execute('select id ' |
|  | 'from cards ' |
|  | 'where number = ? and userId = ?', |
|  | [request.form['number'], session['userId']]) |
|  | card = historCard.fetchone() |
|  |  |
|  | if not card: |
|  | db.execute('insert into cards(number, valid , svv, userId )' |
|  | ' VALUES (?, ?, ?, ? )', |
|  | [request.form['number'], request.form['valid'], |
|  | request.form['svv'], session['userId']]) |
|  |  |
|  | now = datetime.now() |
|  |  |
|  | db.execute( |
|  | 'insert into history(' |
|  | 'productId, quantity, price, date, userId, cardId, basketId) ' |
|  | 'select ' |
|  | 'b.productsId, ' |
|  | 'b.quantity, ' |
|  | 'p.price, ' |
|  | '?, ' |
|  | 'u.id, ' |
|  | 'c.id, ' |
|  | 'b.id ' |
|  | 'from products p , cards c, basket b, users u ' |
|  | 'where p.id=b.productsId ' |
|  | 'and u.id=b.userId ' |
|  | 'and c.userId=u.id ' |
|  | 'and c.number=?' |
|  | 'and u.id=?', |
|  | [now, request.form['number'], session['userId']]) |
|  |  |
|  | db.execute( |
|  | 'delete from basket ' |
|  | 'where userId=?', [session['userId']]) |
|  |  |
|  | db.commit() |
|  |  |
|  | cur = get\_db().execute( |
|  | 'select p.name as name, ' |
|  | 'p.model as model, ' |
|  | 'h.quantity as quantity, ' |
|  | 'p.price as price,' |
|  | 'u.name as customer, ' |
|  | 'h.date as date ' |
|  | 'from products p, users u, history h ' |
|  | 'where h.userId = ? ' |
|  | 'and h.productId = p.id ' |
|  | 'and u.id = h.userId', [session['userId']]) |
|  |  |
|  | history = cur.fetchall() |
|  |  |
|  | return render\_template('add\_bank.html', history=history) |
|  |  |
|  |  |
|  | @app.route('/basket\_user', methods=['GET']) |
|  | def basket\_user(): |
|  | if not session.get('logged\_in'): |
|  | abort(401) |
|  |  |
|  | db = get\_db() |
|  | cur = db.execute('select p.name, p.model, p.price, b.quantity, (p.price \* b.quantity) ' |
|  | 'as cost ' |
|  | 'from basket b, products p ' |
|  | 'where b.productsId = p.id and b.userId=?', [session['userId']]) |
|  |  |
|  | basket = cur.fetchall() |
|  | cur = db.execute('select sum (p.price \* b.quantity) ' |
|  | 'as sum from basket b, products p ' |
|  | 'where p.id=b.productsId ' |
|  | 'and b.userId=?', [session['userId']]) |
|  | sum = (cur.fetchone()[0]) |
|  | db.commit() |
|  |  |
|  | return render\_template('basket\_user.html', basket=basket, sum=sum) |
|  |  |
|  |  |
|  |  |
|  | @app.route('/shopping\_history', methods=['GET']) |
|  | def shopping\_history(): |
|  | if not session.get('logged\_in'): |
|  | abort(401) |
|  |  |
|  | db = get\_db() |
|  |  |
|  | cur = db.execute('select ' |
|  | 'h.date as date, ' |
|  | 'p.name as p\_name, ' |
|  | 'p.model as p\_model, ' |
|  | 'p.price as price, ' |
|  | 'h.quantity as quantity ' |
|  | 'from users u, history h, products p ' |
|  | 'where p.id = h.productId ' |
|  | 'and u.id = h.userId ' |
|  | 'and u.id = ? order by h.date desc ', |
|  | [session['userId']]) |
|  |  |
|  | shopping\_h = cur.fetchall() |
|  |  |
|  | history = {} |
|  |  |
|  | for h in shopping\_h: |
|  | date = datetime.fromisoformat(h['date']).strftime("%Y-%m-%d %H:%M") |
|  | if date in history: |
|  | history[date].append(h) |
|  | else: |
|  | history[date] = [] |
|  | history[date].append(h) |
|  |  |
|  |  |
|  | print(history) |
|  |  |
|  | return render\_template('shopping\_history.html', shopping\_h=history) |
|  |  |
|  |  |
|  | @app.route('/buy') |
|  | def buy(): |
|  | if not session.get('logged\_in'): |
|  | abort(401) |
|  | db = get\_db() |
|  | cur = db.execute( |
|  | 'select p.name, p.model, p.price, b.quantity, (p.price \* b.quantity)' |
|  | ' as cost ' |
|  | 'from basket b, products p ' |
|  | 'where b.productsId = p.id ' |
|  | 'and b.userId=?', [session['userId']]) |
|  |  |
|  | basket = cur.fetchall() |
|  | cur = db.execute( |
|  | 'select sum (p.price \* b.quantity) as sum ' |
|  | 'from products p, basket b ' |
|  | 'where p.id=b.productsId and b.userId=?', |
|  | [session['userId']]) |
|  |  |
|  | sum = (cur.fetchone()[0]) |
|  |  |
|  | db.execute( |
|  | 'insert into customers(' |
|  | 'name, model, price, quantity, customer, date ) ' |
|  | 'select p.name, p.model, p.price, b.quantity, ' |
|  | ' b.userId, datetime() ' |
|  | 'from products p, basket b ' |
|  | 'where p.id=b.productsId and b.userId=? ', |
|  | [session['userId']]) |
|  |  |
|  | db.commit() |
|  | return render\_template('bay\_products.html', basket=basket, sum=sum) |
|  |  |
|  |  |
|  | @app.route('/customers', methods=['GET']) |
|  | def customers(): |
|  | if not session.get('logged\_in'): |
|  | abort(401) |
|  | db = get\_db() |
|  |  |
|  | cur = db.execute( |
|  | 'select c.name as name, ' |
|  | 'c.model as model, ' |
|  | 'c.price as price, ' |
|  | 'c.quantity as quantity, ' |
|  | 'c.customer as customer, ' |
|  | 'u.login\_ as custom\_login,' |
|  | 'u.name as custom\_name, ' |
|  | 'c.date as date ' |
|  | 'from customers c, users u ' |
|  | 'where c.customer = u.id') |
|  |  |
|  | customers = cur.fetchall() |
|  |  |
|  | return render\_template('customers.html', customers=customers) |
|  |  |
|  |  |
|  | """Это представление позволяет пользователю, если он осуществил вход, добавлять |
|  | новые записи. Оно реагирует только на запросы типа POST, а фактическая форма |
|  | отображается на странице list\_product.""" |
|  |  |
|  |  |
|  | @app.route('/add\_user', methods=['GET', 'POST']) |
|  | def add\_user(): |
|  | if request.method == 'POST': |
|  | login = request.form['login\_'] |
|  | password = request.form['password'] |
|  | if login == '' or password == '': |
|  | return render\_template('add\_user.html', error='Invalid registration, the field must be filled!') |
|  | else: |
|  | db = get\_db() |
|  | cur = db.execute( |
|  | 'select login\_ ' |
|  | 'from users ' |
|  | 'where login\_ = ?', |
|  | [login]) |
|  | log = cur.fetchone() |
|  |  |
|  | if log is not None: |
|  | return render\_template('add\_user.html', |
|  | error='Invalid registration, a user with this log already exists!') |
|  |  |
|  | flash('You where registered') |
|  |  |
|  | db = get\_db() |
|  | db.execute( |
|  | 'insert into users (login\_, password, name, surname, tel, email, is\_admin) ' |
|  | 'values (?, ?, ?, ?, ?, ?, 0)', |
|  | [login, password, |
|  | request.form['name'], |
|  | request.form['surname'], |
|  | request.form['tel'], |
|  | request.form['email']]) |
|  | db.commit() |
|  |  |
|  | return render\_template('add\_user.html', error=None) |
|  |  |
|  |  |
|  | @app.route('/add\_bank') |
|  | def to\_buy(): |
|  | if not session.get('logged\_in'): |
|  | abort(401) |
|  |  |
|  | db = get\_db() |
|  | cur = db.execute( |
|  | 'select sum(p.price \* b.quantity) as sum ' |
|  | 'from products p, basket b ' |
|  | 'where p.id=b.productsId and b.userId=?', |
|  | [session['userId']]) |
|  | sum = cur.fetchall() |
|  |  |
|  | return render\_template('add\_bank.html', sum=sum) |
|  |  |
|  |  |
|  | @app.route('/login', methods=['GET', 'POST']) |
|  | def login(): |
|  | print('login') |
|  | error = None |
|  | if request.method == 'POST': |
|  | db = get\_db() |
|  | cur = db.execute( |
|  | 'select id, login\_, password, is\_admin ' |
|  | 'from users ' |
|  | 'where login\_ = ? and password = ?', |
|  | [request.form['username'], |
|  | request.form['password']]) |
|  | user = cur.fetchone() |
|  |  |
|  | if user is None: |
|  | error = 'Invalid username or password!' |
|  | else: |
|  | session['logged\_in'] = True |
|  | session['is\_admin'] = user['is\_admin'] |
|  | session['userId'] = user['id'] |
|  | flash('You were logged in') |
|  |  |
|  | return redirect(url\_for('list\_product')) |
|  |  |
|  | return render\_template('login.html', error=error) |
|  |  |
|  |  |
|  | """Функция выхода, с другой стороны, удаляет обратно этот ключ из сессии. """ |
|  |  |
|  |  |
|  | @app.route('/logout') |
|  | def logout(): |
|  | session.pop('logged\_in', None) |
|  | session.pop('is\_admin', None) |
|  | flash('You where logged out') |
|  | return redirect(url\_for('list\_product')) |
|  |  |
|  |  |
|  | """Наконец, мы просто добавляем строчку в конце файла, которая запускает сервер, если |
|  | мы хотим запустить этот файл как отдельное приложение:""" |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | app.run() |