

МIНIСТЕРСТВО  ОСВIТИ  І  НАУКИ  УКРАЇНИ

НАЦІОНАЛЬНИЙ   ТЕХНІЧНИЙ   УНІВЕРСИТЕТ   УКРАЇНИ

“КИЇВСЬКИЙ  ПОЛІТЕХНІЧНИЙ  ІНСТИТУТ”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 4

з дисципліни “ Основи web-програмування ”

Тема: “ Інтеграція веб-сервера із базами даних.”

|  |  |  |
| --- | --- | --- |
| Виконав(ла)  студент(ка) II курсу  групи КП-52  Сильченко Микита Ігорович  (*прізвище, ім’я, по батькові*) |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Штрафні бали:   |  |  | | --- | --- | | **Термін здачі** | **Оформлення звіту** | |  |  | | Нараховані бали:   |  |  |  | | --- | --- | --- | | **Корект. програм (2 бала)** | **Відп. на теор. питання (1 бал)** | **Відп. на прогр. питання (2 бала)** | |  |  |  | | Сумарний бал:   |  | | --- | |  | |

Київ 2016

**Мета роботи**

Навчитися працювати із базами даних через веб-сервер.

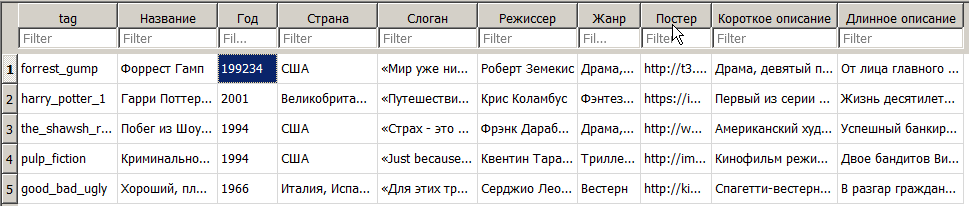
**Постановка завдання**

1. Обрати реляційну або NoSQL базу даних та помістити у неї дані із JSON-файлу ЛР№3.
2. У серверній частині ЛР№3 замінити сховище об’єктів із дискового файлу і списку у оперативній пам’яті на взаємодію зі створеною базою даних.
3. У серверній частині додати JSON REST API для створення, модифікації, отримання (із фільтрацією) і видалення об’єктів.
4. У клієнтській частині додати пагінацію списку об’єктів.

**Лістинг**

|  |
| --- |
| lab4.py |
| **import** sqlite3, os, json, ast **from** flask **import** Flask, g, redirect, url\_for, render\_template, request  app = Flask(\_\_name\_\_)  APP\_ROOT = app.root\_path APP\_STATIC = os.path.join(APP\_ROOT, **'static'**) APP\_DB = os.path.join(APP\_STATIC, **'data/data.db'**)  *#db* **def** connect\_db():  rv = sqlite3.connect(APP\_DB)  rv.row\_factory = sqlite3.Row  **return** rv  **def** get\_db():  **if not** hasattr(g, **'sqlite\_db'**):  g.sqlite\_db = connect\_db()  **return** g.sqlite\_db  @app.teardown\_appcontext **def** close\_db(error):  **if** hasattr(g, **'sqlite\_db'**):  g.sqlite\_db.close()  **def** db\_get\_all\_from\_films():  db = get\_db()  c = db.execute(**"SELECT** *\** **FROM films"**)  **return** c.fetchall()   **def** db\_add\_film(film):  db = get\_db()  **try**:  c = db.execute(**"INSERT INTO films VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"**,  (film[**'tag'**], film[**'Название'**], film[**'Год'**], film[**'Страна'**],  film[**'Слоган'**], film[**'Режиссер'**], film[**'Жанр'**], film[**'Постер'**],  film[**'Короткое описание'**], **""**.join(film[**'Длинное описание'**]))  )  c.fetchall()  db.commit()  **return True  except**:  **return False   def** db\_get\_from\_films(tag):  db = get\_db()  c = db.execute(**"SELECT** *\** **FROM films where tag=(?)"**, (tag,))  **return** c.fetchone()   **def** db\_delete\_from\_films(tag):  db = get\_db()  c = db.execute(**"DELETE FROM films where tag=(?)"**, (tag,))  c.fetchall()  db.commit()   **def** db\_update\_film(tag, film\_update: dict):  film = dict(db\_get\_from\_films(tag))  res\_film = {}  **for** k, v **in** film.items():  **if** isinstance(k, str):  k = k.replace(**'o'**, **'о'**)  res\_film[k] = film\_update[k] **if** film\_update.get(k) **else** v  film\_update[k] = **'spanned'   for** k, v **in** film\_update.items():  **if** v **is not 'spanned'**:  **return False** db = get\_db()   **try**:  c = db.execute(**"""UPDATE films SET "tag" = (?), "Название" = (?),  "Год" = (?), "Страна" = (?),  "Слоган" = (?), "Режиссер" = (?),  "Жанр" = (?), "Постер" = (?),  "Короткое описание" = (?), "Длинное описание" = (?)  where tag=(?)"""**, (res\_film[**'tag'**], res\_film[**'Название'**], res\_film[**'Год'**], res\_film[**'Страна'**],  res\_film[**'Слоган'**], res\_film[**'Режиссер'**], res\_film[**'Жанр'**], res\_film[**'Постер'**],  res\_film[**'Короткое описание'**], **""**.join(res\_film[**'Длинное описание'**]), tag))  **except**:  **return False** c.fetchall()  db.commit()  **return True   def** db\_get\_films\_num():  db = get\_db()  c = db.execute(**"SELECT** *COUNT***(***\****) FROM films"**, ())  **return** c.fetchone() *#//db* @app.route(**'/'**) **def** home():  **return** redirect(url\_for(**"index"**))  @app.route(**'/index.html'**) **def** index():  **return** render\_template(**"index.html"**)  @app.route(**'/about.html'**) **def** about():  **return** render\_template(**"about.html"**)  *# REST API* @app.route(**'/films'**) **def** REST\_films():  films\_list = [dict(film) **for** film **in** db\_get\_all\_from\_films()]  result\_films = []  **for** film **in** films\_list:  **if** ((request.args.get(**'yearlt'**) **and** film[**"Год"**] < int(request.args.get(**'yearlt'**))) **or not** request.args.get(**'yearlt'**)) **and** \  ((request.args.get(**'yeargt'**) **and** film[**"Год"**] > int(request.args.get(**'yeargt'**))) **or not** request.args.get(**'yeargt'**)):  result\_films.append(film)  **return** json.dumps(result\_films, ensure\_ascii=**False**)   @app.route(**'/add\_film'**, methods=[**'POST'**]) **def** REST\_add\_film():  **if** isinstance(request.data, (bytearray, bytes)):  json\_str = request.data.decode(**"utf-8"**)  **else**:  json\_str = request.data  film = ast.literal\_eval(json\_str.strip(**'﻿'**))   **if not** db\_add\_film(film):  returned = {  **"status"**: **"notok"**,  **"error"**: **"occupied tag"** }  **else**:  returned = {  **"status"**: **"ok"**,  **"inserted"**: dict(film)  }   **return** json.dumps(returned, ensure\_ascii=**False**)   @app.route(**'/film/<int:id>'**, methods=[**'GET'**, **'DELETE'**]) **def** REST\_film(id):  **if** db\_get\_films\_num()[0] < id **or** id < 1:  returned = {  **"status"**: **"notok"**,  **"error"**: **"invalid id"** }  **return** json.dumps(returned, ensure\_ascii=**False**)   films\_list = [dict(film) **for** film **in** db\_get\_all\_from\_films()]  **if** request.method == **'GET'**:  **return** json.dumps(films\_list[id - 1], ensure\_ascii=**False**)  **elif** request.method == **'DELETE'**:  db\_delete\_from\_films(films\_list[id - 1][**'tag'**])   returned = {  **"status"** : **"ok"**,  **"deleted"** : dict(films\_list[id - 1])  }  **return** json.dumps(returned, ensure\_ascii=**False**)   @app.route(**'/film/<tag>'**, methods=[**'GET'**, **'DELETE'**]) **def** REST\_film\_by\_tag(tag):  film = db\_get\_from\_films(tag)  **if not** film:  returned = {  **"status"**: **"notok"**,  **"error"**: **"invalid tag"** }  **return** json.dumps(returned, ensure\_ascii=**False**)   **if** request.method == **'GET'**:  **return** json.dumps(dict(film), ensure\_ascii=**False**)  **elif** request.method == **'DELETE'**:  db\_delete\_from\_films(tag)   returned = {  **"status"**: **"ok"**,  **"deleted"**: dict(film)  }  **return** json.dumps(returned, ensure\_ascii=**False**)   @app.route(**'/update\_film/<int:id>'**, methods=[**'POST'**]) **def** REST\_update\_film(id):  **if** db\_get\_films\_num()[0] < id **or** id < 1:  returned = {  **"status"**: **"notok"**,  **"error"**: **"invalid id"** }   **if** isinstance(request.data, (bytearray, bytes)):  json\_str = request.data.decode(**"utf-8"**)  **else**:  json\_str = request.data  film\_update = ast.literal\_eval(json\_str.strip(**'﻿'**))   films\_list = [dict(film) **for** film **in** db\_get\_all\_from\_films()]  film = films\_list[id - 1]   **if** film\_update.get(**'tag'**) **and** (dict(db\_get\_from\_films(film\_update[**'tag'**])) != film):  returned = {  **"status"**: **"notok"**,  **"error"**: **"occupied tag"** }  **return** json.dumps(returned, ensure\_ascii=**False**)   **if not** db\_update\_film(film[**'tag'**], film\_update):  returned = {  **"status"**: **"notok"**,  **"error"**: **"invalid keyvals "** }  **else**:  returned = {  **"status"**: **"ok"**,  **"updated"**: dict(film)  }   **return** json.dumps(returned, ensure\_ascii=**False**)   @app.route(**'/update\_film/<tag>'**, methods=[**'POST'**]) **def** REST\_update\_film\_by\_tag(tag):  **if** isinstance(request.data, (bytearray, bytes)):  json\_str = request.data.decode(**"utf-8"**)  **else**:  json\_str = request.data  film\_update = ast.literal\_eval(json\_str.strip(**'﻿'**))   film = db\_get\_from\_films(tag)  **if** film\_update.get(**'tag'**) **and** (db\_get\_from\_films(film\_update[**'tag'**]) != film):  returned = {  **"status"**: **"notok"**,  **"error"**: **"occupied tag"** }  **return** json.dumps(returned, ensure\_ascii=**False**)   **if not** db\_update\_film(tag, film\_update):  returned = {  **"status"**: **"notok"**,  **"error"**: **"invalid keyvals"** }  **else**:  returned = {  **"status"**: **"ok"**,  **"updated"**: dict(film)  }   **return** json.dumps(returned, ensure\_ascii=**False**) *# //REST API  #* @app.route(**'/films\_table.html'**) **def** films\_table():  films = db\_get\_all\_from\_films()   **return** render\_template(**'films\_table.html'**, films=films)  *#* @app.route(**'/films\_descr/<tag>.html'**) **def** film\_full\_descr(tag):  film = db\_get\_from\_films(tag)   **return** render\_template(**'full\_descr.html'**, film=film)   **if** \_\_name\_\_ == **'\_\_main\_\_'**:  app.run() |

**Структура бази даних**



**REST документація**

|  |  |  |
| --- | --- | --- |
| **Метод** | **URL** | **Дія** |
| GET | /films | Повертає усі фільми |
| GET | /film/<int:id> | Повертає фільм з ідентифікатором id(по пагінації в клієнтській частині) |
| GET | /film/<tag> | Повертає фільм з тегом tag |
| POST | /add\_film | Додає новий фільм |
| POST | /update\_film/<int:id> | Обновлює фільм з ідентифікатором id(по пагінації в клієнтській частині) |
| POST | /update\_film/<tag> | Обновлює фільм з тегом tag |
| DELETE | /film/<int:id> | Видалає фільм з ідентифікатором id(по пагінації в клієнтській частині) |
| DELETE | /film/<tag> | Видаляє фільм з тегом tag |