**Лабораторна робота №6**

**Вебсервіси**

Мета роботи:

– Ознайомитися з технологіями для реалізації веб-сервісів.

– Створити програму для роботи з веб-сервісом.

* 1. Порядок виконання роботи

Тексти класів сервісу зображено в лістингу 1.2.

Текст класу сервіс серверної зображено в лістингу 1.2

Лістинг 1.2

package service;

import java.util.List;

import java.util.Map;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.QueryParam;

import javax.ws.rs.core.MediaType;

import controller.Controller;

import controller.DbConnector;

@Path("/lab\_6\_Olenchenko")

public class Lab6Service {

@GET

@Path("/test")

@Produces(MediaType.TEXT\_PLAIN)

public String test() {

return "Test done!";

}

@GET

@Path("/exist/{table}")

@Produces(MediaType.TEXT\_PLAIN)

public String tableExist(

@PathParam("table") String table,

@QueryParam("db") String db ) {

DbConnector.setDbFullName(db);

Boolean b = Controller.tableExist(table.toUpperCase());

return b.toString();

}

@GET

@Path("/execute/")

@Produces(MediaType.TEXT\_PLAIN)

public String execute(

@QueryParam("db") String db,

@QueryParam("query") String query) {

DbConnector.setDbFullName(db);

List<Map<String, Object>> mapList =

Controller.executeQuery(query);

return mapList.toString();

}

@GET

@Path("/create/{table}")

@Produces(MediaType.TEXT\_PLAIN)

public int createTable (

@PathParam("table") String table,

@QueryParam("db") String db) {

DbConnector.setDbFullName(db);

int b = Controller.createTable(table.toUpperCase());

return b;

}

}

Текст класу Controller клієнтської частини зображено в лістингу 1.3

Лістинг 1.3

package controller;

import java.lang.reflect.Method;

import java.net.URISyntaxException;

import java.util.LinkedHashMap;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.Map;

import org.apache.http.HttpEntity;

import org.apache.http.HttpResponse;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.client.methods.HttpRequestBase;

import org.apache.http.client.utils.URIBuilder;

import org.apache.http.impl.client.HttpClientBuilder;

import org.apache.http.util.EntityUtils;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.util.List;

public class Controller {

@SuppressWarnings("unused")

private static List<Map<String, Object>> stringToMapList(String result) {

List<Map<String, Object>> list = new ArrayList<>();

try {

JSONArray ar = new JSONArray(result);

for (int i = 0; i < ar.length(); i++) {

JSONObject obj = (JSONObject) ar.get(i);

Map<String, Object> map = new LinkedHashMap<>();

Iterator<String> itr = obj.keys();

while (itr.hasNext()) {

String key = itr.next();

map.put(key, obj.get(key));

}

list.add(map);

}

} catch (JSONException e) {

e.printStackTrace();

}

return list;

}

private static HttpClient httpClient = HttpClientBuilder.create().build();

private static URIBuilder bd;

public static String executeRequest(HttpRequestBase request) {

try {

HttpResponse response = httpClient.execute(request);

int code = response.getStatusLine().getStatusCode();

if (code != 200)

return "ERROR! " + response.getStatusLine().toString();

HttpEntity entity = response.getEntity();

return EntityUtils.toString(entity);

} catch (Exception e) {

return "ERROR! " + e.getStackTrace()[0].toString();

}

}

public static boolean tableExist(String tableName) {

String db = RestController.getDbFullName();

String uri = "http://localhost:8080/lab\_6\_Olenchenko"

+ "/rest/lab\_6\_Olenchenko/exist/"+ tableName +"/";

try {

bd = new URIBuilder(uri);

bd.addParameter("db", db);

String res = executeRequest(new HttpGet(bd.build()));

return res != null && res.equals("true");

} catch (URISyntaxException e) {

e.printStackTrace();

return false;

}

}

public static List<Map<String, Object>> executeQuery(String query) {

String db = RestController.getDbFullName();

String uri = "http://localhost:8080/lab\_6\_Olenchenko"

+ "/rest/lab\_6\_Olenchenko/execute/";

String result="";

try {

bd = new URIBuilder(uri);

bd.addParameter("db", db);

bd.addParameter("query", query);

result = executeRequest(new HttpGet(bd.build()));

return stringToMapList(result);

} catch (URISyntaxException e) {

e.printStackTrace();

return new ArrayList<>();

}

}

public static int executeUpdate(String query) {

String db = RestController.getDbFullName();

String uri = "http://localhost:8080/lab\_6\_Olenchenko"

+ "/rest/lab\_6\_Olenchenko/update/";

try {

bd = new URIBuilder(uri);

bd.addParameter("db", db);

bd.addParameter("query", query);

String res = executeRequest(new HttpPost(bd.build())); //Post

return 0;

} catch (Exception e) {

e.printStackTrace();

return 0;

}

}

public static int createTable(final String tableName) {

try {

final String queryClass = "query.Query" + tableName;

final Class<?> clz = Class.forName(queryClass);

final Method mtd = clz.getMethod("queryCreate", (Class<?>[])new Class[0]);

final String sql = (String)mtd.invoke(null, new Object[0]);

final int n = executeUpdate(sql);

return n;

}

catch (Exception e1) {

e1.printStackTrace();

return 0;

}

}

public static void add(final String tableName, final Map<String, Object> map) {

try {

final String queryClass = "query.Query" + tableName;

final Class<?> clz = Class.forName(queryClass);

final Method mtd = clz.getMethod("queryAdd", Map.class);

final String sql = (String)mtd.invoke(null, map);

executeUpdate(sql);

}

catch (Exception e) {

e.printStackTrace();

}

}

public static void edit(final String tableName, final Map<String, Object> map) {

try {

final String queryClass = "query.Query" + tableName;

final Class<?> clz = Class.forName(queryClass);

final Method mtd = clz.getMethod("queryEdit", Map.class);

final String sql = (String)mtd.invoke(null, map);

executeUpdate(sql);

}

catch (Exception e) {

e.printStackTrace();

}

}

public static void delete(final String tableName, final Map<String, Object> map) {

try {

final String queryClass = "query.Query" + tableName;

final Class<?> clz = Class.forName(queryClass);

final Method mtd = clz.getMethod("queryDelById", Integer.TYPE);

final int id = (int) map.get("id");

final String sql = (String)mtd.invoke(null, id);

executeUpdate(sql);

}

catch (Exception e) {

e.printStackTrace();

}

}

}

Текст класу RestController клієнтської частини зображено в лістингу 1.4

Лістинг 1.4

package controller;

import javax.swing.JOptionPane;

public class RestController {

private static String dbFullName = null;

public static void setDbFullName(String dbFllName) {

RestController.dbFullName =dbFllName;

}

public static String getDbFullName() {

if(dbFullName==null) {

dbFullName = JOptionPane.showInputDialog(null,

"Enter DB full name", "d:/dbOlenchenko");

}

return dbFullName;

}

}

## Результати тестових запусків для різних запитів через Postmen і рядок браузера.

На рисунку 1.2 зображено результат виконання test.

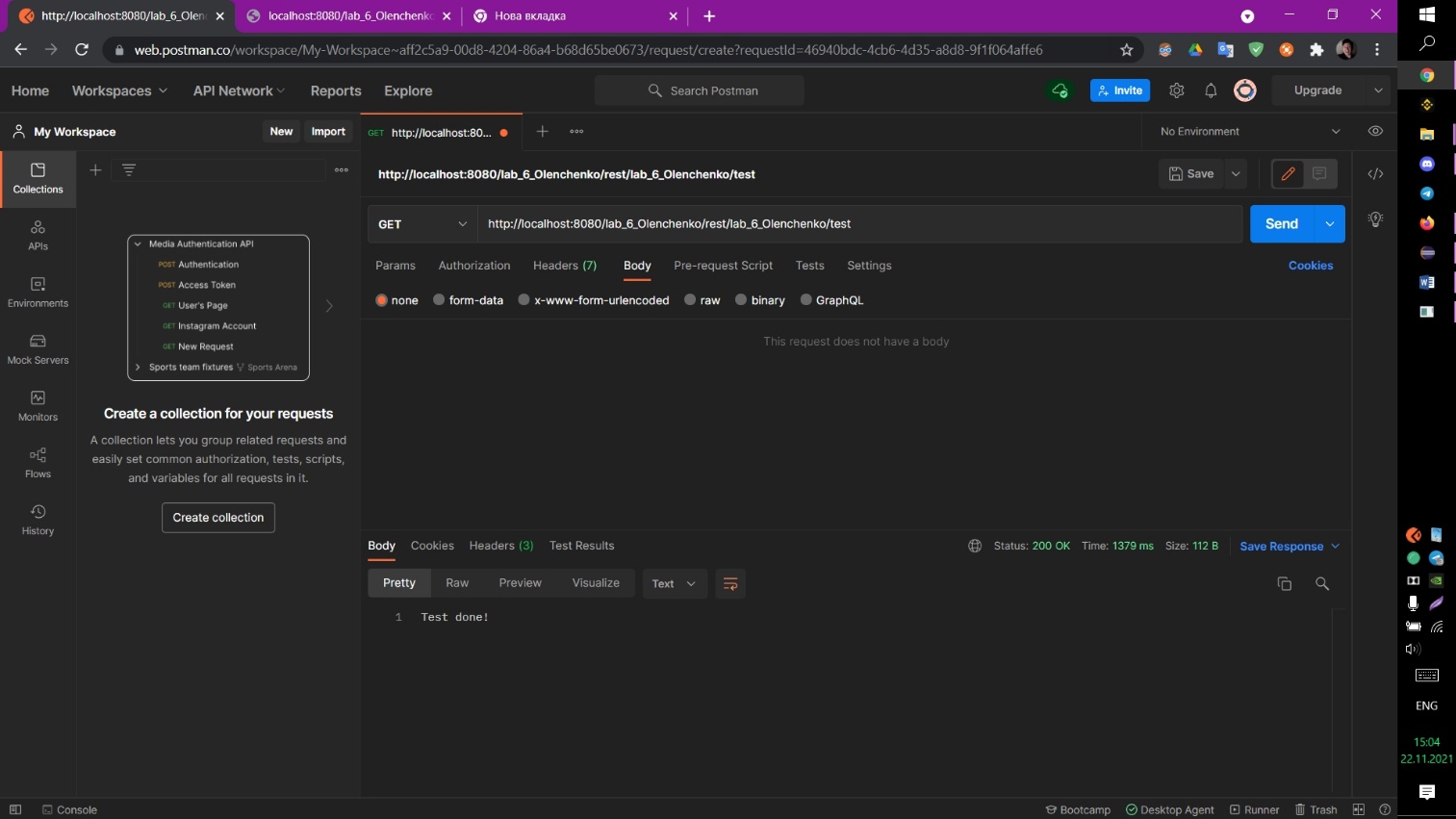


Рисунок 1.2 - результат виконання test.

На рисунку 1.3 зображено результат виконання методу exist

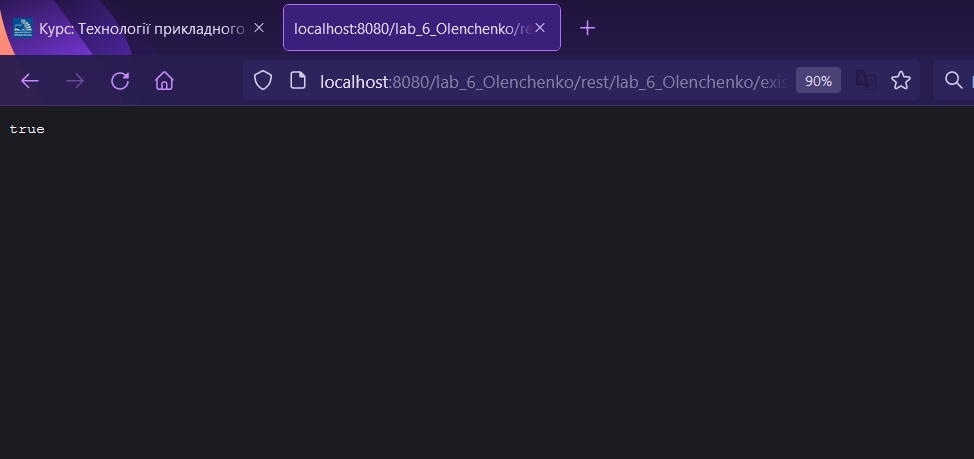


Рисунок 1.3 - результат виконання exist.

На рисунку 1.4 зображено результат виконання методу execute

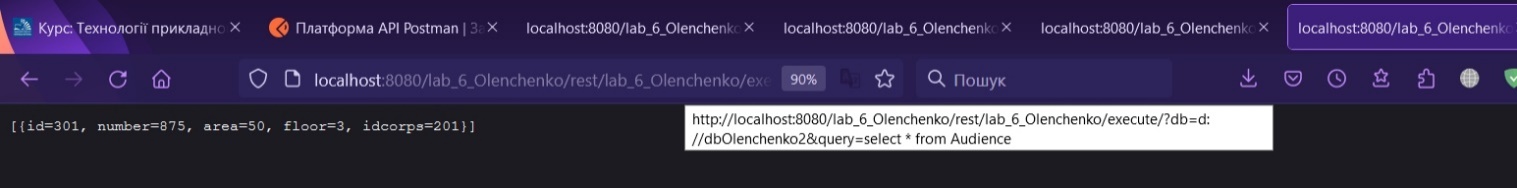


Рисунок 1.4 - Результат виконання execute.

Результати тестування довільних запитів до своєї бази через клієнта зображено на рисунку 1.5.

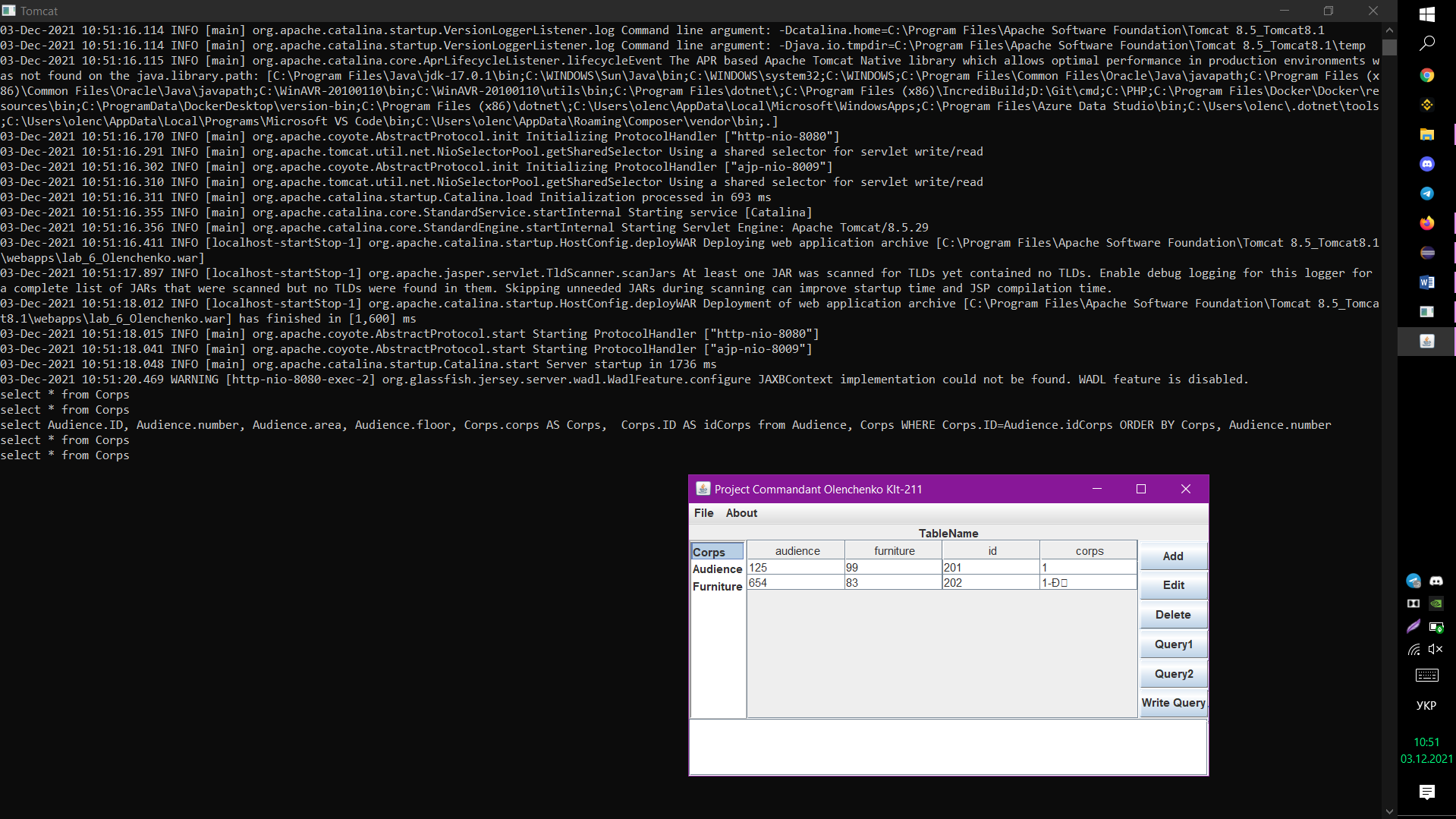


Рисунок 1.5 - Результат бази через клієнта

**Висновок:** Ознайомилися з технологіями для реалізації інтерфейсу користувача. Створили програму для роботи з реляційною базою даних через WEB інтерфейс.