МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Звіт

з практичної роботи № 4 з дисципліни

Програмування під Android

Виконав: Перевірив:

ст. гр. ПЗПІ-23-4 ст. викладач кафедри ПІ

Цисельський Андрій Сокорчук Ігор Петрович

Харків 2024

ЗАВДАННЯ

Ознайомитися із різними способами збереження даних у Android, зокрема з використанням SharedPreferences, бази даних SQLite та роботи з файлами. Навчитися застосовувати ці методи у прикладних проєктах.

ХІД РОБОТИ

Cтворюємо новий проєкт та обираємо Empty Views Activity.

Почнемо з того, що створимо об’єкт, який зберігається:

1 . public class User {

2 . Integer id;

3 . String name;

4 . Integer age;

5 . public User(Integer id, String name, Integer age) {

6 . this.name = name;

7 . this.age = age;

8 . this.id = id;

9 . }

10. public Integer getId() {return id;}

11. public String getName() {return name;}

12. public Integer getAge() {return age;}

13. }

Отже, ми маємо користувача з ім'ям, віком та ID, який буде використовуватися пізніше для роботи з базою даних SQLite. Створимо інтерфейс для взаємодії з користувачем [ДОДАТОК А]. Використовуємо ConstraintLayout та вкладений RecyclerView. Додамо поля для введення імені та віку нового користувача, а також кнопки для збереження та зчитування інформації.

SQLite — це вбудована реляційна база даних, яка дозволяє працювати з великими обсягами структурованих даних без необхідності налаштування сервера. Дані зберігаються локально в файлі.

Для роботи з SQLite у Android зазвичай створюється клас, який успадковується від SQLiteOpenHelper.

Отже, давайте створимо клас DatabaseHelper та занесему в нього логіку роботи з даними [ДОДАТОК Б]. DatabaseHelper — це клас, що надає інтерфейс для роботи з базою даних SQLite та збереження налаштувань користувача за допомогою SharedPreferences. Він успадковується від SQLiteOpenHelper і дозволяє виконувати операції з базою даних, такі як додавання нових користувачів, зчитування користувачів та збереження їх у локальні налаштування.

Основні функції:

1. Збереження даних користувача в базі: Метод insertUser() додає нового користувача до таблиці user в базі даних.
2. Читання даних користувачів: Метод getUsersFromDatabase() витягує всі користувацькі дані з таблиці user і повертає їх у вигляді списку об'єктів User.
3. Збереження та зчитування налаштувань: Дані користувача (ім'я та вік) також зберігаються у SharedPreferences через методи saveUserToPreferences(), getNameFromPreferences() та getAgeFromPreferences().

Цей клас дозволяє зручно зберігати і отримувати як дані користувача в базі даних, так і локальні налаштування для зручності користувача.

Далі створюємо клас UserAdapter [ДОДАТОК В]. Клас UserAdapter є адаптером для RecyclerView, який використовує механізм DataBinding для зручного відображення даних користувачів. Він працює з списком об'єктів User, кожен з яких має ім'я, вік та ID. Адаптер має кілька важливих методів, які визначають, як дані користувачів будуть представлені в інтерфейсі. Метод updateUserList відповідає за оновлення списку користувачів у адаптері. Якщо переданий список не порожній, він очищує поточний список і додає нові елементи, після чого викликається notifyDataSetChanged() для оновлення відображення в RecyclerView. Метод onCreateViewHolder створює нові елементи інтерфейсу, інфлейтячи розмітку з допомогою DataBinding. Він повертає новий об'єкт UserViewHolder, який тримає посилання на ListUserBinding, що відповідає за прив'язку даних до елементів інтерфейсу. Метод onBindViewHolder прив'язує конкретні дані користувача (ім'я та вік) до відповідних полів в інтерфейсі.

Клас також містить метод getItemCount, який повертає кількість елементів у списку, необхідну для відображення. Весь процес забезпечує зручну інтеграцію між даними та UI за допомогою DataBinding, що значно спрощує роботу з інтерфейсом користувача.

Переходимо до MainActivity [ДОДАТОК Г]. MainActivity є головною активністю додатка, що дозволяє користувачам додавати нових користувачів, зберігати та відображати налаштування, а також працювати з базою даних та локальними файлами.

Основний функціонал включає:

1. Додавання користувачів: Користувач може ввести ім'я та вік, і після натискання кнопки ці дані додаються до бази даних SQLite, а також зберігаються в налаштуваннях.
2. Відображення користувачів: За допомогою RecyclerView та адаптера UserAdapter відображається список користувачів, що зберігаються в базі даних.
3. Збереження та зчитування налаштувань: Є можливість зберігати налаштування (у вигляді тексту) у файл та зчитувати їх для відображення.
4. Локальні файли: Дані з налаштувань записуються у файл config.txt, який згодом можна прочитати.

ВИСНОВКИ

Під час практичної роботи було вивчено основні методи збереження даних в Android: використання SharedPreferences для зберігання простих налаштувань користувача у вигляді пар ключ-значення, застосування SQLite для роботи з великими структурованими даними в реляційній базі, а також робота з файлами для запису та зчитування даних з локального сховища додатка.

Посилання на відеозвіт завантажений на YouTube. URL: <https://youtu.be/zwR8eOr0wEc>

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Документація AndroidDevelopers SharedPreferences. URL: https://developer.android.com/training/data-storage/shared-preferences (дата звернення: 15.11.2024)
2. Документація AndroidDevelopers Save data using SQLite. URL: https://developer.android.com/training/data-storage/sqlite (дата звернення: 15.11.2024)
3. Документація AndroidDevelopers SQLiteDatabase. URL: https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase (дата звернення: 15.11.2024)

ДОДАТОК А

Файл розмітки activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools">  
 <androidx.constraintlayout.widget.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:padding="16dp"  
 android:id="@+id/main">  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/configLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/config"  
 android:hint="Enter config"  
 android:inputType="text"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/write"  
 android:text="Save Config"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toBottomOf="@id/configLayout"  
 app:layout\_constraintStart\_toStartOf="parent"  
 android:layout\_marginTop="16dp" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/read"  
 android:text="Load Config"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toBottomOf="@id/write"  
 app:layout\_constraintStart\_toStartOf="parent"  
 android:layout\_marginTop="8dp" />  
  
 <TextView  
 android:id="@+id/result"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="8dp"  
 android:padding="8dp"  
 android:background="@android:color/darker\_gray"  
 android:textColor="@android:color/white"  
 app:layout\_constraintTop\_toBottomOf="@id/read"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 tools:text="Config output will appear here" />  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/nameLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toBottomOf="@id/result"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:layout\_marginTop="16dp">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/name"  
 android:hint="Name"  
 android:inputType="text"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.textfield.TextInputLayout  
 android:id="@+id/ageLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toBottomOf="@id/nameLayout"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 android:layout\_marginTop="8dp">  
  
 <com.google.android.material.textfield.TextInputEditText  
 android:id="@+id/age"  
 android:hint="Age"  
 android:inputType="text"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
 </com.google.android.material.textfield.TextInputLayout>  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/add"  
 android:text="Add User"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toBottomOf="@id/ageLayout"  
 app:layout\_constraintStart\_toStartOf="parent"  
 android:layout\_marginTop="16dp" />  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/list"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:layout\_marginTop="8dp"  
 app:layout\_constraintTop\_toBottomOf="@id/add"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"  
 tools:listitem="@layout/list\_user" />  
  
 </androidx.constraintlayout.widget.ConstraintLayout>  
  
</layout>

ДОДАТОК Б

Код програми DatabaseHelper.java

package tsyselskyi.andrii.nure;  
  
import android.annotation.SuppressLint;  
import android.content.ContentValues;  
import android.content.Context;  
import android.content.SharedPreferences;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.util.Log;  
  
import androidx.annotation.Nullable;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class DatabaseHelper extends SQLiteOpenHelper {  
 private static final String *DATABASE\_NAME* = "app.db";  
 private static final int *DATABASE\_VERSION* = 1;  
  
 private static final String *TABLE\_USER* = "user";  
 private static final String *COLUMN\_ID* = "id";  
 private static final String *COLUMN\_NAME* = "name";  
 private static final String *COLUMN\_AGE* = "age";  
  
 private static final String *SQL\_CREATE\_TABLE\_USER* =  
 "CREATE TABLE IF NOT EXISTS " + *TABLE\_USER* + " (" +  
 *COLUMN\_ID* + " INTEGER PRIMARY KEY, " +  
 *COLUMN\_NAME* + " TEXT, " +  
 *COLUMN\_AGE* + " INTEGER)";  
  
  
 private static final String *PREFS\_NAME* = "app\_prefs";  
 private static final String *PREF\_KEY\_NAME* = "name";  
 private static final String *PREF\_KEY\_AGE* = "age";  
  
 private final SharedPreferences sharedPreferences;  
 private final SQLiteDatabase database;  
  
 public DatabaseHelper(@Nullable Context context) {  
 super(context, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  
 this.sharedPreferences = context.getSharedPreferences(*PREFS\_NAME*, Context.*MODE\_PRIVATE*);  
 this.database = getWritableDatabase();  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 db.execSQL(*SQL\_CREATE\_TABLE\_USER*);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
 }  
  
 public void saveUserToPreferences(String name, int age) {  
 sharedPreferences.edit()  
 .putString(*PREF\_KEY\_NAME*, name)  
 .putInt(*PREF\_KEY\_AGE*, age)  
 .apply();  
 }  
  
 public String getNameFromPreferences() {  
 return sharedPreferences.getString(*PREF\_KEY\_NAME*, "");  
 }  
  
 public int getAgeFromPreferences() {  
 return sharedPreferences.getInt(*PREF\_KEY\_AGE*, 0);  
 }  
  
  
 public void insertUser(User user) {  
 ContentValues values = new ContentValues();  
 values.put(*COLUMN\_NAME*, user.getName());  
 values.put(*COLUMN\_AGE*, user.getAge());  
  
 long rowId = database.insert(*TABLE\_USER*, null, values);  
 if (rowId == -1) {  
 Log.*e*("DatabaseHelper", "Failed to insert user");  
 }  
 }  
  
 @SuppressLint("Range")  
 public List<User> getUsersFromDatabase() {  
 List<User> users = new ArrayList<>();  
  
 try (Cursor cursor = database.query(  
 *TABLE\_USER*, null, null, null, null, null, null)) {  
  
 while (cursor.moveToNext()) {  
 int id = cursor.getInt(cursor.getColumnIndex(*COLUMN\_ID*));  
 String name = cursor.getString(cursor.getColumnIndex(*COLUMN\_NAME*));  
 int age = cursor.getInt(cursor.getColumnIndex(*COLUMN\_AGE*));  
  
 users.add(new User(id, name, age));  
 }  
  
 } catch (Exception e) {  
 Log.*e*("DatabaseHelper", "Error reading users", e);  
 }  
  
 return users;  
 }  
}

ДОДАТОК В

Код програми UserAdapter.java

package tsyselskyi.andrii.nure;  
  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
import androidx.annotation.NonNull;  
import androidx.databinding.DataBindingUtil;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.ArrayList;  
import java.util.List;  
  
import tsyselskyi.andrii.nure.databinding.ListUserBinding;  
  
public class UserAdapter extends RecyclerView.Adapter<UserAdapter.UserViewHolder> {  
 private List<User> usersList = new ArrayList<>();  
  
 public void updateUserList(List<User> newUsers) {  
 if (newUsers != null && !newUsers.isEmpty()) {  
 usersList.clear();  
 usersList.addAll(newUsers);  
 notifyDataSetChanged();  
 }  
 }  
  
 @NonNull  
 @Override  
 public UserViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 ListUserBinding binding = DataBindingUtil.*inflate*(  
 LayoutInflater.*from*(parent.getContext()),  
 R.layout.*list\_user*,  
 parent,  
 false  
 );  
 return new UserViewHolder(binding);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull UserViewHolder holder, int position) {  
 User currentUser = usersList.get(position);  
 holder.binding.setNameString(currentUser.getName());  
 holder.binding.setAgeString(String.*valueOf*(currentUser.getAge()));  
 }  
  
 @Override  
 public int getItemCount() {  
 return usersList.size();  
 }  
  
 public static class UserViewHolder extends RecyclerView.ViewHolder {  
 final ListUserBinding binding;  
  
 public UserViewHolder(@NonNull ListUserBinding binding) {  
 super(binding.getRoot());  
 this.binding = binding;  
 }  
 }  
}

ДОДАТОК Г

Код програми MainActivity.java

package tsyselskyi.andrii.nure;  
  
import android.os.Bundle;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.databinding.DataBindingUtil;  
import androidx.recyclerview.widget.LinearLayoutManager;  
  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.io.OutputStreamWriter;  
import java.util.List;  
  
import tsyselskyi.andrii.nure.databinding.ActivityMainBinding;  
  
public class MainActivity extends AppCompatActivity {  
  
 private UserAdapter adapter;  
 private ActivityMainBinding binding;  
 private DatabaseHelper databaseHelper;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 binding = DataBindingUtil.*setContentView*(this, R.layout.*activity\_main*);  
 databaseHelper = new DatabaseHelper(this);  
  
 setupRecyclerView();  
 setupListeners();  
  
 loadUserFromPreferences();  
 }  
  
 private void setupRecyclerView() {  
 adapter = new UserAdapter();  
 binding.list.setLayoutManager(new LinearLayoutManager(this));  
 binding.list.setAdapter(adapter);  
 updateUserList();  
 }  
  
 private void setupListeners() {  
 binding.write.setOnClickListener(v -> saveConfig());  
 binding.read.setOnClickListener(v -> displayConfig());  
 binding.add.setOnClickListener(v -> addUser());  
 }  
  
 private void saveConfig() {  
 String configData = binding.config.getText().toString().trim();  
 if (!configData.isEmpty()) {  
 writeToFile(configData);  
 binding.config.setText("");  
 showToast("Config saved!");  
 } else {  
 showToast("Config is empty!");  
 }  
 }  
  
 private void displayConfig() {  
 binding.result.setText(readFromFile());  
 }  
  
 private void addUser() {  
 String name = binding.name.getText().toString().trim();  
 String ageStr = binding.age.getText().toString().trim();  
  
 if (name.isEmpty() || ageStr.isEmpty()) {  
 showToast("Name or Age cannot be empty");  
 return;  
 }  
  
 try {  
 int age = Integer.*parseInt*(ageStr);  
 User user = new User(null, name, age);  
 databaseHelper.insertUser(user);  
  
 databaseHelper.saveUserToPreferences(name, age);  
  
 updateUserList();  
  
 binding.name.setText("");  
 binding.age.setText("");  
  
 showToast("User added!");  
 } catch (NumberFormatException e) {  
 showToast("Invalid age format");  
 }  
 }  
  
 private void loadUserFromPreferences() {  
 String name = databaseHelper.getNameFromPreferences();  
 int age = databaseHelper.getAgeFromPreferences();  
  
 if (!name.isEmpty() && age != 0) {  
 binding.name.setText(name);  
 binding.age.setText(String.*valueOf*(age));  
 }  
 }  
  
 private void updateUserList() {  
 List<User> users = databaseHelper.getUsersFromDatabase();  
 adapter.updateUserList(users);  
 }  
  
 private void writeToFile(String data) {  
 try (OutputStreamWriter writer = new OutputStreamWriter(openFileOutput("config.txt", *MODE\_PRIVATE*))) {  
 writer.write(data);  
 } catch (Exception e) {  
 e.printStackTrace();  
 showToast("Failed to save config");  
 }  
 }  
  
 private String readFromFile() {  
 StringBuilder content = new StringBuilder();  
 try (BufferedReader reader = new BufferedReader(new InputStreamReader(openFileInput("config.txt")))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 content.append(line).append("\n");  
 }  
 } catch (Exception e) {  
 e.printStackTrace();  
 showToast("Failed to read config");  
 }  
 return content.toString().trim();  
 }  
  
 private void showToast(String message) {  
 Toast.*makeText*(this, message, Toast.*LENGTH\_SHORT*).show();  
 }  
}