**ЛЕКЦІЯ 15. РОБОТА З НАЛАШТУВАННЯМИ ТА СТАНОМ ДОДАТКУ**

**10.1 Збереження та відновлення стану Activity**

Клас *Activity* має в своєму складі методи, призначені для збереження/відновлення стану екземпляра *Activity*:

protected void onRestoreInstanceState(Bundle saveInstanceState);

protected void onSaveInstanceState(Bundle saveInstanceState);

Обидва ці методи в якості параметра приймають об’єкт Bundle, який і містить стан activity.

В якій ситуації може бути доречно використання подібних методів? Найпоширеніша ситуація – переворот екрану і перехід від портретної орієнтації до альбомної і навпаки. Якщо, наприклад, графічний інтерфейс містить текстове поле *TextView*, і ми програмно змінюємо його текст, то після зміни орієнтації екрану його текст може зникнути. Або якщо у нас глобальні змінні, то при зміні орієнтації екрану їх значення можуть бути скинуті до значень за замовчуванням.

Розглянемо приклад. Нехай маємо файл *activity\_main* такого змісту:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/android)" android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical">

<EditText android:id=["@+id](mailto:@+id)/nameBox" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Введіть ім’я"/>

<Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Зберегти" android:onClick="saveName"/>

<TextView

android:id=["@+id](mailto:@+id)/nameView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:textSize="18dip"/>

<Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Отримати ім’я" android:onClick="getName"/>

</LinearLayout>

Тут визначено поле *EditText*, в яке вводимо ім’я. Також визначена кнопка для його збереження. Для подальшого виведення збереженого імені призначене поле *TextView*, а для отримання збереженого імені – інша кнопка.

Тепер змінимо клас *MainActivity* таким чином:

public class MainActivity extends AppCompatActivity { String name="undefined";

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

public void saveName(View view) {

// отримуємо введене ім’я

EditText nameBox=(EditText) findViewById(R.id.nameBox);

name=nameBox.getText().toString();

}

public void getName(View view) {

// отримуємо збережене ім’я

TextView nameView=(TextView) findViewById(R.id.nameView);

nameView.setText(name);

}

}

Для зберігання імені в програмі визначена змінна *name*. При натисканні на першу кнопку зберігаємо текст з *EditText* в змінну name, а при натисканні на другу кнопку – отримуємо текст з змінної name в поле *TextView*.

Запустимо додаток, введемо ім’я, збережемо і отримаємо його в *TextView* (рис. 10.1). Та якщо ми перейдемо до альбомного режиму, то *TextView* виявиться порожнім. І навіть якщо ми спробуємо заново отримати значення з змінної name, то ми побачимо, що вона обнулилася (рис. 10.2).

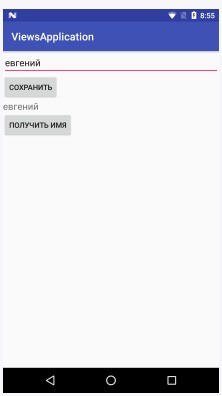


Рисунок 10.1 Вікно додатку при вертикальній орієнтації пристрою

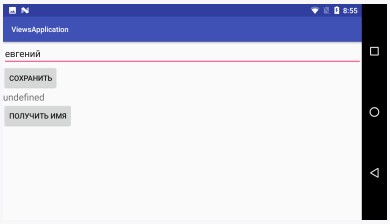


Рисунок 10.2 Вікно додатку при горизонтальній орієнтації пристрою

Щоб уникнути подібних ситуацій слід зберігати і відновлювати стан activity. Для цього змінимо код MainActivity наступним чином:

public class MainActivity extends AppCompatActivity {

String name="undefined";

final static String nameVariableKey="NAME\_VARIABLE";

final static String textViewTexKey="TEXTVIEW\_TEXT";

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

// збереження стану

[@Override](mailto:@Override)

protected void onSaveInstanceState(Bundle outState) {

outState.putString(nameVariableKey, name);

TextView nameView=(TextView) findViewById(R.id.nameView);

outState.putString(textViewTexKey, nameView.getText().toString());

super.onSaveInstanceState(outState);

}

// отримання раніше збереженого стану

[@Override](mailto:@Override)

protected void onRestoreInstanceState(Bundle savedInstanceState) {

super.onRestoreInstanceState(savedInstanceState);

name = savedInstanceState.getString(nameVariableKey);

String textViewText= savedInstanceState.getString(textViewTexKey); TextView nameView = (TextView) findViewById(R.id.nameView);

nameView.setText(textViewText);

}

public void saveName(View view) {

// отримуємо введене ім’я

EditText nameBox=(EditText) findViewById(R.id.nameBox);

name=nameBox.getText().toString();

}

public void getName(View view) {

// отримуємо збережене ім’я

TextView nameView=(TextView) findViewById(R.id.nameView);

nameView.setText(name);

}

}

У методі *onSaveInstanceState*() зберігаємо стан. Для цього викликаємо у

параметра *Bundle* метод *putString(key, value)*, перший параметр якого – ключ, а другий – значення для збереження даних. В даному випадку ми зберігаємо два рядки, тому викликаємо метод *putString*(). Для збереження об’єктів інших типів даних ми можемо викликати відповідний метод:

 *putInt();*

 *putDouble();*

 *putFloat();*

 *putChar(;)*

 *putByte().*

Кожен із перерахованих методів в якості першого параметра приймає ключ, а в якості другого – значення.

В методі *onRestoreInstanceState()* відбувається зворотний процес – за допомогою методу *getString(key)* по ключу отримуємо із збереженого стану рядок по ключу. Відповідно для отримання даних інших типів ми можемо використовувати аналогічні методи:

 *getInt();*

 *getDouble();*

 *getFloat();*

 *getChar();*

 *getByte().*

**10.2 Збереження та отримання налаштувань**

**10.2.1 Загальні принципи роботи із налаштуваннями**

Часто в програмі доводиться зберігати невеликі шматочки даних для подальшого використання (дані про користувача, налаштування конфігурації і т. д.).

Для цього в Android існує концепція *Preferences*. Налаштування представляють собою групу пар ключ-значення, які використовуються додатком.

В якості значень можуть виступати дані наступних типів: *Boolean*, *Float*, *Integer*,

*Long*, *String*, набір рядків.

Налаштування є загальними для всіх activity в додатку, але також можуть бути і налаштування безпосередньо для окремих activity

Налаштування зберігаються в xml-файлах в незашифрованому вигляді в локальному сховищі. Вони невидимі, тому для простого користувача недоступні.

При роботі з налаштуваннями слід враховувати наступні моменти. оскільки вони зберігаються в незашифрованому вигляді, то не рекомендується зберігати в них чутливі дані типу пароля або номерів кредитних карт. Крім того, вони представляють дані, що асоціюються з додатком, і використовуючи панель управління додатком в налаштуваннях ОС користувач може видалити ці дані.

**10.2.2 Загальні налаштування**

Для роботи із загальними налаштувапннями в класі *Activity* (точніше, в його базовому класі *Context*) є метод *getSharedPreferences*():

import android.content.SharedPreferences;

//...........................

SharedPreferences settings = getSharedPreferences("PreferencesName", MODE\_PRIVATE);

Перший параметр методу вказує на назву налаштувань. В даному випадку це "*PreferencesName*". Якщо налаштувань з подібною назвою немає, то вони створюються при виклику даного методу. Другий параметр вказує на режим доступу. В даному випадку режим описаний константою *MODE\_PRIVATE*.

Клас *android.content.SharedPreferences* надає ряд методів для управління налаштуваннями:

 *contains(String key)*: повертає true, якщо в налаштуваннях збережено значення із ключем *key*;

 *getAll()*: повертає всі збережені в налаштуваннях значення;

 *getBoolean(String key, boolean defValue)*: повертає з налаштувань значення типу *Boolean*, якому відповідає ключ *key*. Якщо елемента з таким ключем не виявиться, то повертається значення *defValue*, передане іншим параметром;

 *getFloat(String key, float defValue)*: повертає значення типу *float* з ключем *key*.

Якщо елемента з таким ключем не виявиться, то повертається значення *defValue*;

 *getInt(String key, int defValue)*: повертає значення типу *int* з ключем *key*;

 *getLong(String key, long defValue)*: повертає значення типу *long* з ключем *key*;

 *getString(String key, String defValue)*: повертає стрічкове значення з ключем

*key*;

 *getStringSet(String key, Set <String> defValues)*: повертає масив рядків з

ключем *key*;

 *edit()*: повертає об’єкт *SharedPreferences*.Editor, який використовується для редагування налаштувань.

Для управління налаштуваннями використовується об’єкт класу

*SharedPreferences*.Editor, що повертається методом *edit()*. Він має такі методи:

 *clear()*: витирає всі налаштування;

 *remove(String key)*: витирає із налаштувань значення з ключем *key*;

 *putBoolean(String key, boolean value)*: додає в налаштування значення типу

*boolean* з ключем *key*;

 *putFloat(String key, float value)*: додає в налаштування значення типу *float* з ключем *key*;

 *putInt(String key, int value)*: додає в налаштування значення *int* з ключем *key*;

 *putLong(String key, long value)*: додає в налаштування значення типу *long* з ключем *key*;

 *putString(String key, String value)*: додає в настройки рядок з ключем *key*;

 *putStringSet (String key, Set <String> values)*: додає в налаштування масив стрічок;

 *commit()*: підтверджує всі зміни в налаштуваннях;

 *apply()*: також, як і метод *commit()*, підтверджує всі зміни в налаштуваннях, проте змінений об’єкт *SharedPreferences* спочатку зберігається в тимчасовій пам’яті, і лише потім в результаті асинхронної операції записується на мобільний пристрій.

Розглянемо приклад збереження та отримати налаштувань додатком. Визначимо в файлі activity\_main.xml наступний призначений для користувача інтерфейс:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical">

<EditText android:id=["@+id](mailto:@+id)/nameBox" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:hint="Введіть ім’я"/>

<Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Зберегти" android:onClick="saveName"/>

<TextView android:id=["@+id](mailto:@+id)/nameView" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" />

<Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Отримати ім’я" android:onClick="getName"/>

</LinearLayout>

На екрані будуть дві кнопки – для збереження і для виведення попередньо збереженого значення, а також поле для введення і текстове поле для виведення збереженого налаштування. Визначимо методи обробники кнопок в класі *MainActivity*: public class MainActivity extends AppCompatActivity {

private static final String PREFS\_FILE = "Account"; private static final String PREF\_NAME = "Name"; SharedPreferences settings;

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

settings = getSharedPreferences(PREFS\_FILE, MODE\_PRIVATE);

}

public void saveName(View view) {

EditText nameBox = (EditText) findViewById(R.id.nameBox); String name = nameBox.getText().toString();

// Зберігаємо введено ім’я в налаштуваннях SharedPreferences.Editor prefEditor = settings.edit(); prefEditor.putString(PREF\_NAME, name);

prefEditor.apply();

}

public void getName(View view) {

// отримуємо збережене ім’я

TextView nameView = (TextView) findViewById(R.id.nameView); String name = settings.getString(PREF\_NAME,"не определено");

nameView.setText(name);

}

}

За відсутності налаштувань при спробі їх отримати, додаток виведе значення за

замовчуванням. Написаний код дає можливість зберегти і вивести на екран збережене значення.

Інколи виникає потреба автоматично зберігати дані, що вводяться при виході користувача з activity. Для цього ми можемо перевизначити метод *onPause*():

public class MainActivity extends AppCompatActivity { private static final String PREFS\_FILE="Account"; private static final String PREF\_NAME="Name"; EditText nameBox;

SharedPreferences settings; SharedPreferences.Editor prefEditor;

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

nameBox=(EditText) findViewById(R.id.nameBox);

settings=getSharedPreferences(PREFS\_FILE, MODE\_PRIVATE);

// отримуємо налаштування

String name=settings.getString(PREF\_NAME,"");

nameBox.setText(name);

}

[@Override](mailto:@Override)

protected void onPause(){

super.onPause();

EditText nameBox=(EditText) findViewById(R.id.nameBox); String name=nameBox.getText().toString();

// зберігаємо в налаштуваннях prefEditor=settings.edit(); prefEditor.putString(PREF\_NAME, name);

prefEditor.apply();

}

public void saveName(View view) {

}

public void getName(View view) {

}

}

**10.2.3 Приватні налаштування**

Крім загальних налаштувань кожна activity може використовувати приватні, доступ до яких з інших activity буде неможливим. Для отримання налаштувань рівня activity використовується метод *getPreferences(MODE\_PRIVATE)*:

import android.content.SharedPreferences;

//........................

SharedPreferences settings = getPreferences(MODE\_PRIVATE);

Тобто, на відміну від загальних налаштувань, тут не використовується назва групи налаштувань в якості першого параметра, так як це відбувається в методі *getSharedPreferences()*. Однак вся інша робота по додаванню, отриманню та зміні налаштувань буде аналогічною розглянутій вище роботі із загальними налаштуваннями.

**10.3 PreferenceFragment**

Для спрощення роботи з групою налаштувань Android надає спеціальний тип фрагмента – *PreferenceFragment*. Розглянемо як його можна використовувати. Створимо новий проект і спочатку додамо в папку *res* підпапку *xml*. Потім в папку *res / xml* додамо новий файл, який назвемо *settings.xml* із таким вмістом:

<?xml version="1.0" encoding="utf-8"?>

<PreferenceScreen xmlns:andr[oid="http://schemas.android.com/apk/res/android">](http://schemas.android.com/apk/res/android)

<EditTextPreference android:key="login" android:summary="Введіть логін" android:title="Логін" />

<CheckBoxPreference android:key="enabled" android:summary="Показати логін" android:title="Показати" />

</PreferenceScreen>

Тут в кореневому елементі *PreferenceScreen* встановлюються елементи *EditTextPreference* і *CheckBoxPreference*. Через кожен з цих елементів ми можемо взаємодіяти з певною налаштуванням. В даному випадку ми можемо використовувати декілька різних типів налаштувань:

 *EditTextPreference*: використовується елемент *EditText* для введення текстового значення;

 *CheckBoxPreference*: використовується елемент *CheckBox* для встановлення логічних значень *true* або *false*;

 *SwitchPreference*: використовується елемент *Switch* для встановлення логічних значень *true* або *false* ("on" і "off");

 *RingtonePreference*: використовує діалогове вікно для встановлення рінгтона із списку рингтонів для встановлення логічних значень *true* або *false*;

 *ListPreference*: використовує список для вибору одного з визначених значень;

 *MultiSelectListPreference*: також використовує список для вибору значень, але дозволяє вибрати кілька елементів.

Для кожного елемента налаштування необхідно визначити, як мінімум, три атрибути:

1) *android: key*: задає ключ налаштування в *SharedPreferences*;

2) *android: title*: назва настройки для користувача;

3) *android: summary*: задає короткий опис налаштування для користувача. Нарешті додамо в проект спеціальну activity для задання налаштувань. Назвемо

її *SettingsActivity*:

public class SettingsActivity extends AppCompatActivity {

[@Override](mailto:@Override)

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_settings);

getFragmentManager()

.beginTransaction()

.add(R.id.prefs\_content, new SettingsFragment())

.commit();

}

public static class SettingsFragment extends PreferenceFragment {

[@Override](mailto:@Override)

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

addPreferencesFromResource(R.xml.settings);

}

}

}

*SettingsActivity* в якості розмітки інтерфейсу буде використовувати ресурс *R.layout.activity\_settings*. При запуску *SettingsActivity* буде завантажувати фрагмент *SettingsFragment* в елемент з ідентифікатором *prefs*\_content.

Сам фрагмент *SettingsFragment* успадковується від класу *PreferenceFragment*. В його методі *onCreate()* викликається метод addPreferencesFromResource(), в який передається *id* ресурсу *xml* з налаштуваннями (в даному випадку раніше визначений ресурс *R.xml.settings*).

Тепер визначимо в папці *res / layout* наступний файл *activity\_settings.xml*:

<?xml version="1.0" encoding="utf-8"?>

<FrameLayout xmlns:andr[oid="http://schemas.android.com/apk/res/android](http://schemas.android.com/apk/res/android)" android:id=["@+id](mailto:@+id)/prefs\_content"

android:layout\_width="match\_parent" android:layout\_height="match\_parent" />

Тут визначено *FrameLayout* з ідентифікатором prefs\_*content* - саме той елемент, в який буде завантажуватися фрагмент *SettingsFragment*.

Тепер перейдемо до головної activity - *MainActivity*. У файлі *activity\_main.xml*

опишемо текстове поле і кнопку:

<LinearLayout xmlns:android=["http://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) android:layout\_width="match\_parent" android:layout\_height="match\_parent"

android:orientation="vertical">

<TextView android:id=["@+id](mailto:@+id)/loginText" android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:textSize="18sp" />

<Button android:layout\_width="match\_parent" android:layout\_height="wrap\_content" android:text="Налаштування" android:onClick="setPrefs"/>

</LinearLayout>

І нарешті, змінимо клас *MainActivity*:

public class MainActivity extends AppCompatActivity {

TextView loginText; boolean enabled; String login;

[@Override](mailto:@Override)

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.activity\_main);

loginText=(TextView) findViewById(R.id.loginText);

}

[@Override](mailto:@Override)

public void onResume() {

super.onResume();

SharedPreferences prefs=PreferenceManager.getDefaultSharedPreferences(this);

enabled = prefs.getBoolean("enabled", false);

login = prefs.getString("login", "не установлено");

loginText.setText(login);

if(enabled)

loginText.setVisibility(View.VISIBLE);

else loginText.setVisibility(View.INVISIBLE);

}

public void setPrefs(View view){

Intent intent=new Intent(this, SettingsActivity.class);

startActivity(intent);

}

}

Тут в методі *onResume*() ми отримуємо всі налаштування. Якщо налаштування

*enabled* дорівнює *true*, то відображаємо текстове поле з логіном.

У методі *setPrefs()*, який спрацьовує при натисканні на кнопку, відбувається перехід до *SettingsActivity*.

При першому запуску налаштувань не буде, і логін не буде відображуватися. Перейдемо на сторінку налаштувань, встановимо там логін і включимо його відображення, а потім повернемося на головну activity (рис. 10.3).

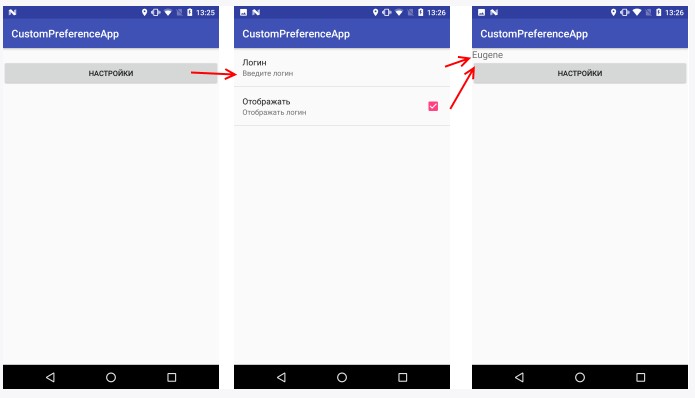


Рисунок 10.3 Демонстрація автоматичного збереження налаштувань додатку

Тобто, вручну нам нічого не треба зберігати, всі налаштування автоматично зберігаються функціоналом *PreferenceFragment*.