Тема 11. Локальні та захищені сховища мобільної ОС. Робота з файлами та збереження користувацьких налаштувань

У той час як можливості зберігання та обробки мобільних пристроїв збільшуються, телефони та планшети, як і раніше, відстають від своїх настільних комп'ютерів та ноутбуків колег. Тому варто запланувати архітектуру сховища даних для вашої програми, а не просто припускати, що база даних є правильною відповіддю весь час. Існує кілька різних варіантів, які відповідають різним вимогам, наприклад:

- Налаштування – Android пропонує вбудований механізм зберігання простих пар «ключ-значення» даних. Якщо ви зберігаєте прості параметри користувача або невеликі фрагменти даних (наприклад, відомості про персоналізацію), використовуйте власні функції платформи для збереження таких відомостей.
- Текстові файли – вхідні дані користувача або кеші вмісту, що завантажується (наприклад, HTML) може зберігатися безпосередньо у файловій системі. Використовуйте відповідну угоду про назву файлів, щоб упорядкувати файли та знайти дані.
- Серіалізовані файли даних – об'єкти можуть зберігатися як XML або JSON у файловій системі. Платформа .NET включає бібліотеки, які спрощують серіалізацію та десеріалізацію об'єктів. Використовуйте відповідні імена для впорядкування файлів даних.
- База даних – ядро СУБД SQLite доступне на платформі Android і корисне для зберігання структурованих даних, які необхідно вимагати, сортувати або іншим чином обробляти. Сховище бази даних підходить для списків даних з багатьма властивостями.
- Файли зображень – хоча двійкові дані можна зберігати в базі даних на мобільному пристрої, рекомендується зберігати їх безпосередньо у файловій системі. Якщо потрібно, можна зберегти імена файлів у базі даних, щоб зв'язати зображення з іншими даними. При роботі з

великими зображеннями або великою кількістю зображень рекомендується спланувати стратегію кешування, яка видаляє файли, які не потрібні, щоб уникнути використання всього дискового простору користувача.

Якщо база даних є правильним механізмом зберігання для програми, у решті цього документа розглядається використання SQLite на платформі Xamarin.

Існує ряд переваг використання бази даних SQL у мобільному додатку:

- Бази даних SQL дозволяють ефективно зберігати структуровані дані.
- Визначені дані можна отримати за допомогою складних запитів.
- Результати запиту можна відсортувати.
- Результати запиту можна агрегувати.
- Розробники з існуючими навичками баз даних можуть використовувати свої знання для розробки коду доступу до бази даних і даних.
- Модель даних із серверного компонента підключеного додатка може використовуватися повторно (загалом або частково) у мобільному додатку.

SQLite – це ядро СУБД з відкритим кодом, яке було прийнято Google для своєї мобільної платформи. Ядро СУБД SQLite вбудовано в обидві операційні системи, тому розробники не можуть скористатися нею додатковими можливостями. SQLite добре підходить для кросплатформної розробки мобільних додатків, так як:

- Ядро СУБД є невеликим, швидким і легко переносимим.
- База даних зберігається в одному файлі, який легко управляти на мобільних пристроях.
- Формат файлу легко використовувати на різних платформах: будь то 32-розрядні або 64-розрядні та великі або маленькі системи.
- Він реалізує більшу частину стандарту SQL92.

Оскільки SQLite призначений для невеликих і швидких, є деякі застереження щодо його використання:

- Деякий синтаксис OUTER не підтримується.
- Підтримуються лише таблиці RENAME та ADDCOLUMN. Ви не можете вносити інші зміни до схеми.
- Подання доступні лише для читання.

Зберігання пар ключ-значення

Якщо потрібен швидкий спосіб зберегти кілька рядків або чисел, слід розглянути можливість використання файлу налаштувань . Дії та служби Android можуть використовувати метод getDefaultSharedPreferences() класу PreferenceManager щоб отримати посилання на об'єкт SharedPreferences, який можна використовувати як для читання, так і для запису у файл налаштувань за замовчуванням.

```
SharedPreferences myPreferences
= PreferenceManager.getDefaultSharedPreferences(MyActivity.this);
```

Щоб розпочати запис у файл налаштувань, необхідно викликати метод edit() об'єкта SharedPreferences, який повертає об'єкт SharedPreferences.Editor.

```
SharedPreferences.Editor myEditor = myPreferences.edit();
```

Об'єкт SharedPreferences.Editor має кілька інтуїтивно зрозумілих методів, які можна використовувати для збереження нових пар ключ-значення у файлі налаштувань. Наприклад, можна використовувати метод putString() для приміщення пари ключ-значення, значення якої має тип String . Так само ви можете використовувати метод putFloat() для приміщення пари ключ-значення, значення якої має тип float . Наступний фрагмент коду створює три пари ключ-значення:

```
myEditor.putString(“NAME”, “Alice”);
myEditor.putInt("AGE", 25);
myEditor.putBoolean("SINGLE?", true);
```

Після додавання всіх пар необхідно викликати метод commit() об'єкта SharedPreferences.Editor щоб вони зберігалися.

```
myEditor.commit();
```

Читання з об'єкта SharedPreferences набагато простіше. Все, що потрібно зробити, це викликати відповідний метод get*() . Наприклад, щоб отримати пару ключ-значення, значення якої має тип String, необхідно викликати метод getString(). Ось фрагмент коду, який повертає всі значення, які ми додали:

```
String name = myPreferences.getString(“NAME”, “unknown”);
int age = myPreferences.getInt("AGE", 0);
boolean isSingle = myPreferences.getBoolean("SINGLE?", false);
```

Як видно з наведеного вище коду, як другий параметр усі методи get*() очікують значення за замовчуванням, тобто значення, яке має бути повернено, якщо ключ відсутній у файлі налаштувань.

Зверніть увагу, що файли налаштувань обмежуються лише рядками та примітивними типами даних. Якщо ви хочете зберігати складніші типи даних або двійкові дані, ви повинні вибрати інший варіант зберігання.

Використання бази даних SQLite

Кожна програма для Android може створювати та використовувати бази даних SQLite для зберігання великих обсягів структурованих даних. Як ви, можливо, вже знаєте, SQLite не тільки легкий, а й дуже швидкий. Якщо у вас є досвід роботи з системами управління реляційними базами даних, і ви знайомі як із SQL (скорочення від мови структурованих запитів), так і з JDBC (скорочення від Java Database Connectivity), це може бути вашим варіантом зберігання.

Щоб створити нову базу даних SQLite або відкрити вже існуючу, можна використовувати openOrCreateDatabase() метод всередині своєї діяльності або служби. Як аргументи ви повинні передати ім'я вашої бази даних та режим, у якому ви хочете її відкрити. Найбільш використовуваний режим – MODE_PRIVATE, який забезпечує доступ до бази даних лише для вашої програми. Наприклад, як ви можете відкрити або створити базу даних з ім'ям my.db:

```
SQLiteDatabase myDB =
```

```
openOrCreateDatabase("my.db", MODE_PRIVATE, null);
```

Після того, як база даних створена, ви можете використовувати метод execSQL() для запуску операторів SQL на ній. У наступному коді показано, як використовувати оператор SQL CREATE TABLE для створення таблиці з ім'ям user , яка має три стовпці:

```
myDB.execSQL(
"CREATE TABLE IF NOT EXISTS user (name VARCHAR(200),
age INT, is_single INT)");
```

Хоча можна вставити нові рядки в таблицю за допомогою execSQL(), краще замість цього використовувати метод insert(). Метод insert() чекає на об'єкт ContentValues, що містить значення для кожного стовпця таблиці. Об'єкт ContentValues дуже схожий на об'єкт Map і містить пари ключ-значення.

Ось два об'єкти ContentValues можна використовувати з user таблицею:

```
ContentValues row1 = New ContentValues();
row1.put("name", "Alice");
row1.put("age", 25);
row1.put("is_single", 1);
ContentValues row2 = New ContentValues();
row2.put("name", "Bob");
row2.put("age", 20);
row2.put("is_single", 0);
```

Як ви вже здогадалися, ключі, які передаються методу put() повинні збігатися з іменами стовпців у таблиці.

Коли ваші об'єкти ContentValues будуть готові, можна передати їх методу insert() разом з ім'ям таблиці.

```
myDB.insert("user", null, row1);
myDB.insert("user", null, row2);
```

Для запиту бази даних ви можете використовувати метод rawQuery() , який повертає об'єкт Cursor, що містить результати запиту.

```
Cursor myCursor =
  myDB.rawQuery("select name, age, is_single from user", null);
```

Об'єкт Cursor може містити нуль або більше рядків. Найпростіший спосіб moveToNext() усі його рядки – викликати метод moveToNext() усередині циклу while.

Щоб отримати значення окремого стовпця, ви повинні використовувати такі методи, як getString() та getInt() , які очікують на індекс стовпця. Наприклад, як ви повинні отримати всі значення, які ви вставили в user таблицю:

```
while(myCursor.moveToNext()) {
String name = myCursor.getString(0);
int age = myCursor.getInt(1);
boolean isSingle = (myCursor.getInt(2)) == 1 ?
}
```

Після того, як ви отримали всі результати вашого запиту, переконайтеся, що ви викликаєте метод close() об'єкта Cursor для вивільнення всіх ресурсів, які він містить.

```
myCursor.close();
```

Так само, коли ви завершили всі операції з базою даних, не забудьте викликати метод close() об'єкта SQLiteDatabase .

```
myDB.close();
```

Використання внутрішнього сховища

Кожна Android-програма має власний внутрішній каталог зберігання, в якому вона може зберігати текстові та двійкові файли. Файли цього каталогу недоступні користувачеві або іншим програмам, встановленим на пристрої користувача. Вони також автоматично видаляються, коли користувач видаляє програму.

Перш ніж ви зможете використовувати каталог внутрішнього сховища, ви повинні визначити його розташування. Для цього можна викликати метод getFilesDir() , який доступний як для дій, так і для служб.

```
File internalStorageDir = getFilesDir();
```

Щоб отримати посилання на файл усередині каталогу, ви можете передати ім'я файлу разом із певним місцезнаходженням. Наприклад, як ви можете отримати посилання на файл з ім'ям alice.csv :

```
File alice = new File(internalStorageDir, "alice.csv");
```

З цього моменту ви можете використовувати свої знання класів та методів введення/виведення Java для читання або запису у файл. У наступному фрагменті коду показано, як використовувати об'єкт FileOutputStream та його метод write() для запису у файл:

```
// Create file output stream
fos = new FileOutputStream(alice);
// Write a line to the file
fos.write(“Alice,25,1”.getBytes());
// Close the file output stream
fos.close();
```

Використання зовнішнього сховища

Оскільки обсяг внутрішньої пам'яті пристроїв Android зазвичай фіксований і дуже обмежений, деякі пристрої Android підтримують зовнішні носії, такі як знімні карти micro-SD. Я рекомендую використовувати цю опцію зберігання для великих файлів, таких як фотографії та відео.

На відміну від внутрішнього сховища, зовнішнє сховище може бути не завжди доступним. Тому ви завжди повинні перевіряти, чи він змонтований перед використанням. Для цього використовуйте метод getExternalStorageState() класу Environment.

```
if(Environment.getExternalStorageState()
        .equals(Environment.MEDIA_MOUNTED)) {
    // External storage is usable
} else {
    // External storage is not usable
    // Try again later
}
```

Переконавшись, що зовнішнє сховище є доступним, ви можете отримати шлях до каталогу зовнішнього сховища для вашої програми, викликавши метод getExternalFilesDir() і передавши йому значення null як аргумент. Потім можна використовувати шлях для посилання на файли всередині каталогу. Наприклад, ось як ви можете послатися на файл bob.jpg у зовнішньому каталозі вашої програми:

```
File bob = new File(getExternalFilesDir(null), "bob.jpg");
```

WRITE_EXTERNAL_STORAGE користувача надати вам дозвіл WRITE_EXTERNAL_STORAGE , ви можете отримати доступ на читання / запис до всієї файлової системи у зовнішньому сховищі. Потім можна використовувати загальнодоступні загальнодоступні каталоги для зберігання фотографій, фільмів та інших медіафайлів. Клас Environment пропонує метод getExternalStoragePublicDirectory() визначення шляхів цих загальнодоступних каталогів.

Наприклад, передавши метод значення Environment. DIRECTORY_PICTURES, ви можете визначити шлях до загальнодоступного каталогу, в якому ви можете зберігати фотографії. Аналогічно, якщо ви передаєте методу значення Environment.DIRECTORY_MOVIES , ви отримаєте шлях до загальнодоступного каталогу, в якому можна зберігати фільми.

Ось як ви можете послатися на файл з ім'ям bob.jpg у каталозі загальнодоступних зображень:

```
File bobInPictures = new File(
    Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES),
    "bob.jpg"
);
```

Отримавши об'єкт File , ви можете знову використовувати FileInputStream і FileOutputStream для читання або запису в нього.