ЛЕКЦІЯ 13. РОБОТА З БАЗАМИ ДАНИХ, ГРАФІКОЮ І АНІМАЦІЄЮ. РОЗРОБКА ІГОР

13.1 ВВЕДЕННЯ

Будемо рухатися далі в розгляді питань пов'язаних з розробкою додатків для смартфонів. Сучасне програмування важко уявити без використання баз даних, рано чи пізно в процесі розвитку додатки з'являється усвідомлення необхідності довготривалого зберігання і обробки структурованої інформації. Дана лекція присвячена розгляду питань, пов'язаних з використанням баз даних SQLite в додатках, що розробляються під Android. Бази даних SQLite є основою побудови робочої і функціональної програми, в якій необхідно працювати з великими обсягами структурованої інформації.

13.2 ОСНОВИ РОБОТИ З БАЗАМИ ДАНИХ, SQLITE

SQLite - невелика і при цьому потужна система управління базами даних. Ця система створена в 2000 році, її розробник доктор Річард Хіпп (Dr. Richard Hipp). В даний час є однією з найпоширеніших SQL-систем управління базами даних у світі. Можна виділити кілька причин такої популярності SQLite: вона безкоштовна; вона маленька, приблизно 150 Кбайт; не вимагає установки і адміністрування. Детальніше див. .

База даних SQLite - це звичайний файл, його можна переміщати і копіювати на іншу систему (наприклад, з телефону на робочий комп'ютер) і вона буде відмінно працювати. Android зберігає файл бази даних програми у папці:

data / data / packagename / databases /,

де packagename - ім'я пакету, в якому розташоване додаток.

Для доступу до цього файлу необхідно запускати команди SQL, Android за допомогою допоміжних класів і зручних методів приховує частину деталей, але все таки необхідно мати хоча б мінімальні знання про SQL, щоб користуватися цими інструментами.

Звернення до бази даних SQL виконуються за допомогою запитів, існує три основних види SQL запитів: DDL, Modification і Query.

- **DDL запити.** Такі запити використовуються для створення таблиць. Кожна таблиця характеризується ім'ям і описом стовпців, яке містить ім'я шпальти і тип даних. У файлі бази даних може бути декілька таблиць.

Приклад запиту для створення таблиці:

```
create Table_Name (
  _id integer primary key autoincrement,
  field_name_1 text,
  field_name_2 text);
```

Перший стовпець позначений, як primary key (первинний ключ), тобто унікальне число, яке однозначно ідентифікує рядок. Слово autoincrement вказує, що база даних буде автоматично збільшувати значення ключа при додаванні кожного запису, що і забезпечує його унікальність. Існує домовленість перший стовпець завжди називати _id, це не жорстка вимога SQLite, проте може знадобитися при використанні контент-провайдера в Android.

Варто мати на увазі, що в SQLite, на відміну від багатьох інших баз даних, типи даних стовпців є лише підказкою, тобто не викличе ніяких нарікань спроба записати рядок у стовпець, призначений для зберігання цілих чисел або навпаки. Цей факт можна розглядати, як особливість бази даних, а не як помилку, на це звертають увагу автори SQLite.

- **Modification запити.** Такі запити використовуються для додавання, зміни або видалення записів.

Приклад запиту на додавання рядки:

```
insert into Table_Name values (null, value1, value2);
```

У цьому випадку значення розмістяться у відповідні стовпчики таблиці, перше значення задається для поля _id і одно null, SQLite обчислює значення цього поля самостійно.

При додаванні можна вказувати стовпці, в які будуть розміщуватися значення, інші стовпці заповняться значеннями за замовчуванням, в цьому випадку можна додавати елементи в зміненому порядку. Приклад такого запиту:

```
insert into Table_Name (field_name_2, field_name_1)
 values (value2, value1);
```

У цьому випадку додаються значення тільки в поля field_name_1 і field_name_2, причому змінено порядок проходження полів, а разом з цим і порядок проходження значень, іноді це буває зручно. Приклади запитів на зміну рядки:

```
update Table_Name set Field_Name_1 = value;
```

поміняє значення стовпця Field_Name_1 на value у всій таблиці;

```
update Table_Name set Field_Name_1 = value where _id = smth;
```

Поміняє значення стовпця Field_Name_1 тільки в тому рядку, _id якої дорівнює smth. Приклади запитів на видалення рядків:

```
delete from Table_Name;
delete from Table_Name where Field_Name_1 = smth;
```

перший запит видаляє таблицю цілком, другий - тільки ті рядки, в яких стовпець Field_Name_1 має значення smth.

- **Query запити.** Такі запити дозволяють отримувати вибірки з таблиці за різними критеріями. Приклад запиту:

```
select * from Table_Name where (_id = smth);
select Field_Name_1, Field_Name_2 from Table_Name
                                Field_Name_1 = smth);
```

Перший запит виводить рядок з _id рівним smth, другий - виводить два елементи Field_Name_1 і Field_Name_2 рядків, в яких Field_Name_1 дорівнює smth.

Повернемося до розгляду питань, пов'язаних з використанням бази даних SQLite в додатках під Android. Будь база даних, створена в додатку доступна будь-якому класу додатки, але недоступна з поза. Щоб відкрити доступ до бази даних

іншим програмам необхідно використовувати контент-провайдери (Content Providers).

Для створення та оновлення бази даних в Android передбачений клас SQLiteOpenHelper. При розробці програми, що працює з базами даних, необхідно створити клас - SQLiteOpenHelper, в якому обов'язково реалізувати методи:

onCreate () - Викликається при першому створенні бази даних;

onUpgrade () - Викликається, коли необхідно оновити базу даних.

За бажанням можна реалізувати метод:

onOpen () - Викликається при відкритті бази даних.

У цьому ж класі має сенс оголосити рядкові константи, в яких визначити назви таблиць і стовпців. Отриманий клас подбає про відкриття бази даних, якщо вона існує, або про створення її в іншому випадку, а так само про оновлення бази даних у разі потреби.

В Android передбачений клас для роботи з базою даних SQLite безпосередньо, цей клас називається SQLiteDatabase і містить методи:

openDatabase () - Дозволяє відкрити базу даних;

update () - Дозволяє оновити рядки таблиці бази даних;

insert () - Дозволяє додавати рядки в таблицю бази даних;

delete () - Дозволяє видаляти рядки з таблиці бази даних;

query () - Дозволяє складати запити до бази даних;

execSQL () - Дозволяє виконувати запити до бази даних.

Для додавання нових рядків у таблицю використовується клас ContentValues, кожен об'єкт цього класу представляє собою один рядок таблиці і виглядає як асоціативний масив з іменами стовпців і значеннями, які їм відповідають.

Для отримання результатів запитів до бази даних використовується клас Cursor, об'єкти цього класу посилаються на результуючий набір даних, дозволяють управляти поточною позицією в возвращаемом при запиті наборі даних.

Для надання доступу до даних для інших додатків можна використовувати контент-провайдери (ContentProvider). Будь інформація, керована контент-провайдером адресується за допомогою URI:

content: // authority / path / id

де:

content: // - Стандартний необхідний префікс;

authority - Ім'я провайдера, рекомендується використовувати повне кваліфікаційне ім'я пакета для уникнення конфлікту імен;

path - Віртуальна папка всередині провайдера, яка визначає вид запитуваних даних;

id - Первинний ключ окремої запитаної записи, для запиту всіх записів певного типу цей параметр не вказується.

Контент-провайдери підтримують стандартний синтаксис запитів для читання, зміни, вставки і видалення даних.

13.3 АНІМАЦІЯ

Android надає потужні API для анімації елементів користувальницького інтерфейсу і побудови 2D і 3D зображень.

Платформа Android надає дві системи анімації: анімація властивостей, що з'явилася в Android 3.0, і анімація компонентів інтерфейсу користувача (спадкоємців класу View). Розглянемо докладніше обидві ці системи.

Анімація властивостей (Property Animation). Система анімації властивостей дозволяє визначити анімацію для зміни будь-якої властивості об'єкта, незалежно від того зображується воно на екрані чи ні. Використовуючи цю систему, можна задати наступні характеристики анімації:

- **Тривалість** припускає завдання тривалості часового проміжку виконання анімації, за замовчуванням це значення дорівнює 300 мс.
- **Тимчасова інтерполяція** передбачає обчислення значення властивості в кожен момент часу, як функції від проміжку часу, що пройшов з початку анімації.
- **Кількість повторів і поведінку** визначає необхідність повторення анімації при досягненні кінця заданого часового проміжку, а також кількість повторів у разі потреби. Ця ж характеристика дозволяє задати можливість відтворення у зворотному порядку, якщо ця можливість обрана, то анімація прокручується вперед-назад задане число разів.
- **Група анімацій** дозволяє організувати анімації в деяке безліч і задати режим виконання: одночасно, послідовно безперервно або з деякими затримками.

- **Частота оновлення кадрів** визначає, як часто буде відбуватися зміна кадрів анімації. За замовчуванням оновлення відбувається кожні 10 мс, однак швидкість, з якою додаток зможе оновлювати кадри, зрештою, залежить від завантаженості системи.

Велика частина API системи анімації властивостей знаходиться в пакеті **android.animation.** Також можна використовувати блоки інтерполяції, визначені в пакеті **android.view.animation.**

Клас Animator надає базову структуру для створення анімації. Напряму цей клас зазвичай не використовується, тому що забезпечує мінімальну функціональність, тому найчастіше використовуються класи-спадкоємці, що розширюють можливості класу Animator. Розглянемо основні класи, використовувані для створення анімації властивостей.

- ValueAnimator (нащадок класу Animator). Цей клас є головним оброблювачем розподілу часу для анімації властивостей, а також розраховує значення властивості, призначеного для анімації. Він забезпечує всю основну функціональність: розраховує значення анімації і містить розподілені в часі деталі кожної анімації; містить інформацію про необхідність повторень анімації; містить слухачів, які отримують повідомлення про події поновлення; надає можливість задавати користувача типи для обчислення. У процесі анімації властивостей можна виділити дві частини: обчислення значення властивості, для якого визначається анімація, і присвоєння отриманого значення відповідного полю об'єкта. ValueAnimator не виконує другу частину, тому необхідно стежити за оновленнями значень, що обчислюються в класі ValueAnimator, і змінювати об'єкти, підвладні анімації .
- AnimatorSet (нащадок класу Animator). Надає механізми угруповання анімацій, таким чином, що вони виконуються деяким чином відносно один одного. Можна визначати виконання анімацій одночасно, послідовно і з тимчасовими затримками.

Класи-обчислювачі визначають як обчислювати значення заданих властивостей. Вони отримують: дані про розподіл часу, що надаються класом Animator, початкове і кінцеве значення властивості, після чого на основі цих даних

обчислюють значення властивості, для якого виконується анімація. В системі анімації властивостей існують наступні обчислювачі:

- IntEvaluator для обчислення цілочисельних значень властивостей;
- FloatEvaluator для обчислення речових значень властивостей;
- ArgbEvaluator для обчислення значень кольору в шістнадцятковому представленні;
- TypeEvaluator - інтерфейс, що дозволяє створювати власних обчислювачів.

Інтерполятора визначають за допомогою яких функцій від часу, обчислюються значення властивостей, для яких задається анімація. Інтерполятора визначені в пакеті android .view. animation. Якщо жоден з існуючих інтерполятора не підходить, можна створити власний, реалізувавши інтерфейс **TimeInterpolator.**

Детальніше з системою анімації властивостей можна познайомитися за посиланням: http://developer.android.com/guide/topics/graphics/prop-animation.html .

Анімація компонентів для користувача інтерфейсу. Ця система може бути використана для реалізації анімації перетворень над спадкоємцями класу View. Для розрахунку анімації перетворень використовується наступна інформація: початкова точка, кінцева точка, розмір, поворот та інші загальні аспекти анімації. Анімація перетворень може виконувати серії простих змін вмісту екземпляра класу View. Наприклад, для текстового поля можна переміщати, обертати, розтягувати, зживати текст, якщо визначено фонове зображення, воно повинно змінюватися разом з текстом. Пакет **android.view.animation** надає всі класи, необхідні для реалізації анімації перетворень.

Для завдання послідовності інструкцій анімації перетворень можна використовувати або XML, або Android код. Більш кращим є визначення анімації в XML файлах, розташовуватися ці файли повинні в папці **res / anim /** проекту. XML файл повинен мати єдиний кореневий елемент, це може бути будь-який з окремих елементів: <alpha>, <scale>, <translate>, <rotate>, інтерполятор, або ж елемент <set>, який містить групи цих елементів, у тому числі може містити інші елементи < set>. За замовчуванням інструкції анімації виконуються одночасно, щоб задати послідовне виконання необхідно визначити атрибут startOffset.

Детальніше з системою анімації перетворень можна познайомитися за посиланням: http://developer.android.com/guide/topics/graphics/view-animation.html .

Додатково до розглянутих системам анімації може використовуватися, кадрова анімація, яка реалізується швидкою зміною кадрів, кожен кадр є графічним ресурсом і розташовується в папці **res / drawable /** проекту.

Детальніше з кадровою анімацією можна познайомитися за посиланням: http://developer.android.com/guide/topics/graphics/drawable-animation.html .

13.4 2D І 3D ГРАФІКА

При розробці програми важливо чітко розуміти вимоги до графіку в цьому додатку. Для різних графічних завдань необхідні різні техніки їх вирішення. Далі в лекції розглянемо декілька способів зображення графічних об'єктів в Android.

Полотна й графічні об'єкти. Платформа Android надає API для зображення 2D графіки, який дозволяє зображати на полотні свої графічні об'єкти або змінювати існуючі. Для відображення 2D графіки існують два шляхи:

1. Зобразити графіком або анімацію в елементі користувача інтерфейсу. У цьому випадку графіка управляється процесом відображення ієрархії елементів інтерфейсу. Підходить, коли необхідно відобразити просту графіку, що не вимагає динамічних змін.

2. Зображати графіком безпосередньо на полотні (клас Canvas). У цьому випадку необхідно подбати про виклик методу onDraw (), передаючи його в клас Canvas, або ж про виклик одного з draw ... () методів класу Canvas (наприклад, drawPicture ()). Діючи таким чином, можна управляти анімацією. Цей шлях підходить, коли необхідно постійно перемальовувати вікно програми, наприклад, для відео ігор.

Апаратне прискорення. Починаючи з Android 3.0 (API рівень 11), конвеєр зображення 2D графіки в Android підтримує апаратне прискорення. Це означає, що всі операції малювання на полотні виконуються з використанням GPU. У зв'язку зі збільшенням вимог до ресурсів додаток буде споживати більше RAM. Апаратне прискорення доступно за замовчуванням, якщо цільовий рівень API більше або дорівнює 14, але може бути включено явно. Якщо в додатку використовуються тільки стандартні уявлення і графіка, включення апаратного прискорення не повинно привести до яких-небудь небажаних графічним ефектів. Однак через те, що апаратне прискорення підтримується не всіма операціями 2D графіки, його

включення може порушувати деякі користувальницькі зображення або виклики малювання. Проблеми зазвичай проявляються в невидимості деяких елементів, появі виключень або невірно зображених пікселях. Щоб виправити це, Android дозволяє включати або вимикати апаратне прискорення на різних рівнях: рівень додатки, рівень активності, рівень вікна, рівень елемента інтерфейсу.

OpenGL. Android підтримує високопродуктивну 2D і 3D графіку з використанням відкритої графічної бібліотеки OpenGL, точніше OpenGL ES API. Бібліотека OpenGL є крос-платформних API, який визначає стандартний програмний інтерфейс для апаратного забезпечення, що займається обробкою 3D графіки. OpenGL ES є різновидом OpenGL, призначеної для вбудованих пристроїв. Android підтримує кілька версій OpenGL ES API:

- OpenGL ES 1.0 і 1.1 підтримується Android 1.0 і вище;
- OpenGL ES 2.0 підтримується Android 2.2 (API рівень 8) і вище;
- OpenGL ES 3.0 підтримується Android 4.3 (API рівень 18) і вище.

Підтримка OpenGL ES 3.0 на реальному пристрої вимагає реалізації графічного конвеєра, наданої виробником. Тому пристрій з Android 4.3 і вище може не підтримувати OpenGL ES 3.0.

Детальніше з графікою в Android можна познайомитися по посиланнях:

http://developer.android.com/guide/topics/graphics/2d-graphics.html ;

http://developer.android.com/guide/topics/graphics/opengl.html ;

http://developer.android.com/guide/topics/graphics/hardware-accel.html .