

Introducción

En este documento nos vamos a detener en describir las distintas opciones de acceso a datos que proporciona la plataforma y en cómo podemos realizar las tareas más habituales dentro de este apartado.

La plataforma Android proporciona dos herramientas principales para el almacenamiento y consulta de datos estructurados:

- Bases de Datos SQLite
- Content Providers

En estos próximos artículos nos centraremos en la primera opción, **SQLite**, que abarcará todas las tareas relacionadas con el almacenamiento de los datos propios de nuestra aplicación. El segundo de los mecanismos, los **Content Providers**, que trataremos más adelante, nos facilitarán la tarea de hacer visibles esos datos a otras aplicaciones y, de forma recíproca, de permitir la consulta de datos publicados por terceros desde nuestra aplicación.

SQLite

SQLite es un motor de bases de datos muy popular en la actualidad por ofrecer características tan interesantes como su pequeño tamaño, no necesitar servidor, precisar poca configuración, ser transaccional y por supuesto ser de código libre.

Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una completa API para llevar a cabo de manera sencilla todas las tareas necesarias.

En Android, la forma típica para crear, actualizar, y conectar con una base de datos SQLite será a través de una clase auxiliar llamada `SQLiteOpenHelper`, o para ser más exactos, de una clase propia que derive de ella y que debemos personalizar para adaptarse a las necesidades concretas de nuestra aplicación.

La clase `SQLiteOpenHelper` tiene tan sólo un constructor, que normalmente no necesitaremos sobrescribir, y dos métodos abstractos, `onCreate()` y `onUpgrade()`, que deberemos personalizar con el código necesario para crear nuestra base de datos y para actualizar su estructura respectivamente.

Como ejemplo, nosotros vamos a crear una base de datos muy sencilla llamada `BDUsuarios`, con una sola tabla llamada `Usuarios` que contendrá sólo dos campos: nombre e email. Para ellos, vamos a crear una clase derivada de `SQLiteOpenHelper` que llamaremos `UsuariosSQLiteHelper`, donde sobreescribimos los métodos `onCreate()` y `onUpgrade()` para adaptarlos a la estructura de datos indicada:

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;

public class UsuariosSQLiteHelper extends SQLiteOpenHelper {
    //Sentencia SQL para crear la tabla de Usuarios
    String sqlCreate = "CREATE TABLE Usuarios (codigo INTEGER, nombre TEXT)";

    public UsuariosSQLiteHelper(Context contexto, String nombre,
                                CursorFactory factory, int version) {
        super(contexto, nombre, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //Se ejecuta la sentencia SQL de creación de la tabla
        db.execSQL(sqlCreate);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int versionAnterior, int versionNueva) {
        //NOTA: Por simplicidad del ejemplo aquí utilizamos directamente la opción de
        //      eliminar la tabla anterior y crearla de nuevo vacía con el nuevo formato.
        //      Sin embargo lo normal será que haya que migrar datos de la tabla antigua
        //      a la nueva, por lo que este método debería ser más elaborado.

        //Se elimina la versión anterior de la tabla
        db.execSQL("DROP TABLE IF EXISTS Usuarios");
        //Se crea la nueva versión de la tabla
        db.execSQL(sqlCreate);
    }
}
```

Lo primero que hacemos es definir una variable llamado `sqlCreate` donde almacenamos la sentencia SQL para crear una tabla llamada `Usuarios` con los campos alfanuméricos `nombre` e `email`.

El método `onCreate()` será ejecutado automáticamente por nuestra clase `UsuariosDBHelper` cuando sea necesaria la creación de la base de datos, es decir, cuando aún no exista. Las tareas típicas que deben hacerse en este método serán la creación de todas las tablas necesarias y la inserción de los datos iniciales si son necesarios. En nuestro caso, sólo vamos a crear la tabla `Usuarios` descrita anteriormente.

Para la creación de la tabla utilizaremos la sentencia SQL ya definida y la ejecutaremos contra la base de datos utilizando el método más sencillo de los disponibles en la API de SQLite proporcionada por Android, llamado `execSQL()`. Este método se limita a ejecutar directamente el código SQL que le pasemos como parámetro.

El método `onUpgrade()` se lanzará automáticamente cuando sea necesaria una actualización de la estructura de la base de datos o una conversión de los datos.

Un ejemplo práctico: Imaginemos que publicamos una aplicación que utiliza una tabla con los campos usuario e email (llamémoslo versión 1 de la base de datos). Más adelante, ampliamos la funcionalidad de nuestra aplicación y necesitamos que la tabla también incluya un campo adicional por ejemplo con la edad del usuario (versión 2 de nuestra base de datos). Entonces, para que todo funcione correctamente, la primera vez que ejecutemos la versión ampliada de la aplicación necesitaremos modificar la estructura de la tabla Usuarios para añadir el nuevo campo edad. Pues este tipo de cosas son las que se encargará de hacer automáticamente el método `onUpgrade()` cuando intentemos abrir una versión concreta de la base de datos que aún no exista. Para ello, como parámetros recibe la versión actual de la base de datos en el sistema, y la nueva versión a la que se quiere convertir. En función de esta pareja de datos necesitaremos realizar unas acciones u otras.

En nuestro caso de ejemplo optamos por la opción más sencilla, que es borrar la tabla actual y volver a crearla con la nueva estructura, pero como se indica en los comentarios del código, lo habitual será que necesitemos algo más de lógica para convertir la base de datos de una versión a otra y por supuesto para conservar los datos registrados hasta el momento.

Una vez definida nuestra clase **helper**, la apertura de la base de datos desde nuestra aplicación resulta ser algo de lo más sencillo. Lo primero será crear un objeto de la clase `UsuariosSQLiteHelper` al que pasaremos el contexto de la aplicación (en el ejemplo una referencia a la actividad principal), el nombre de la base de datos, un objeto `CursorFactory` que típicamente no será necesario (en ese caso pasaremos el valor `null`), y por último la versión de la base de datos que necesitamos. La simple creación de este objeto puede tener varios efectos:

- Si la base de datos ya existe y su versión actual coincide con la solicitada simplemente se realizará la conexión con ella.
- Si la base de datos existe pero su versión actual es anterior a la solicitada, se llamará automáticamente al método `onUpgrade()` para convertir la base de datos a la nueva versión y se conectará con la base de datos convertida.
- Si la base de datos no existe, se llamará automáticamente al método `onCreate()` para crearla y se conectará con la base de datos creada.

Una vez tenemos una referencia al objeto `UsuariosSQLiteHelper`, llamaremos a su método `getReadableDatabase()` o `getWritableDatabase()` para obtener una referencia a la base de datos, dependiendo si sólo necesitamos consultar los datos o también necesitamos realizar modificaciones, respectivamente.

Ahora que ya hemos conseguido una referencia a la base de datos (objeto de tipo `SQLiteDatabase`) ya podemos realizar todas las acciones que queramos sobre ella. Para nuestro ejemplo nos limitaremos a insertar 5 registros de prueba, utilizando para ello el método ya comentado `execSQL()` con las sentencias `INSERT` correspondientes. Por último cerramos la conexión con la base de datos llamando al método `close()`.

```
import android.app.Activity;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;

public class AndroidBaseDatos extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        //Abrimos la base de datos 'DBUsuarios' en modo escritura
        UsuariosSQLiteHelper usdbh = new UsuariosSQLiteHelper(this, "DBUsuarios", null, 1);
        SQLiteDatabase db = usdbh.getWritableDatabase();

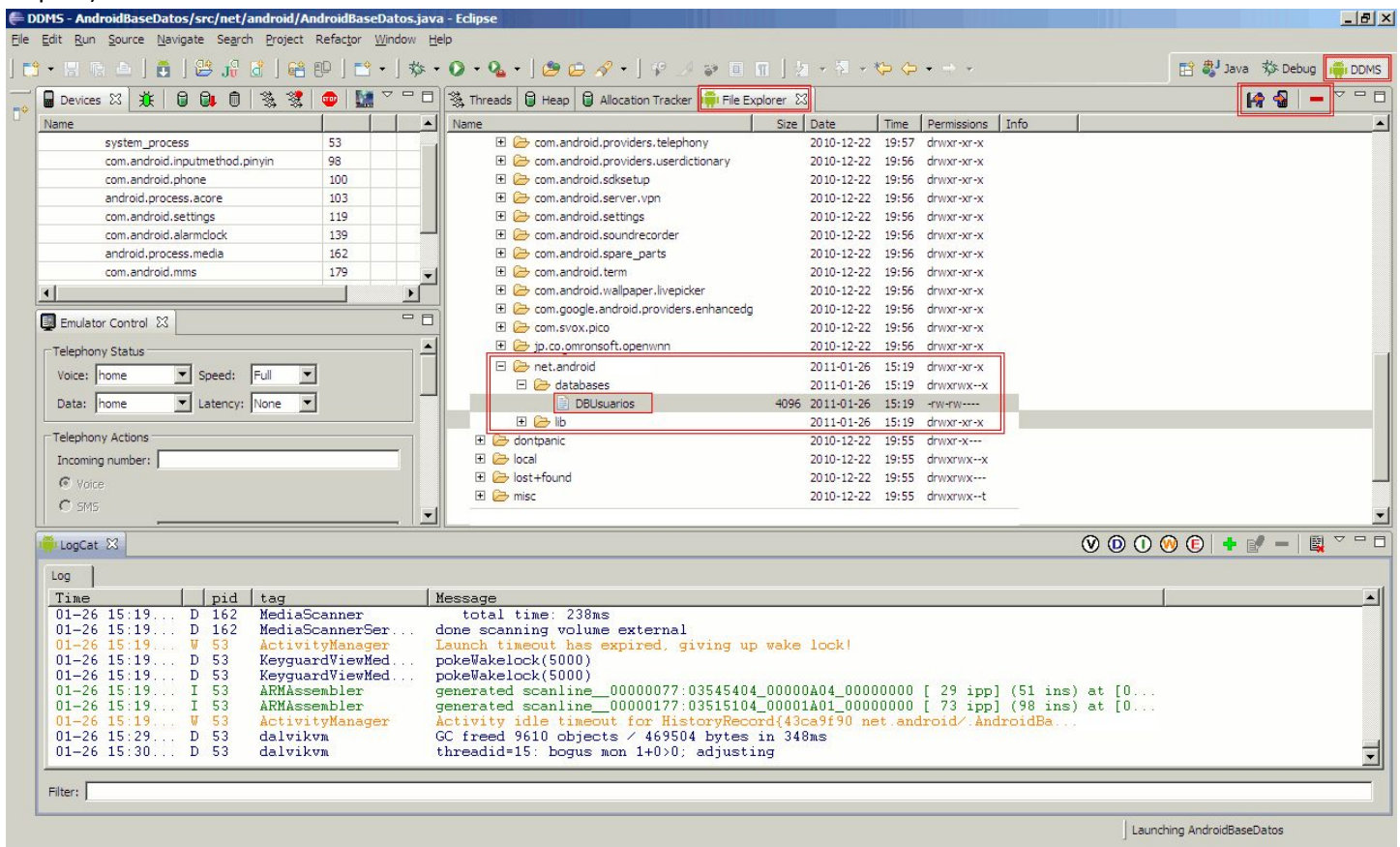
        //Si hemos abierto correctamente la base de datos
        if(db != null) {
            //Insertamos 5 usuarios de ejemplo
            for(int i=1; i<=5; i++) {
                //Generamos los datos
                int codigo = i;
                String nombre = "Usuario" + i;
                //Insertamos los datos en la tabla Usuarios
                db.execSQL("INSERT INTO Usuarios (codigo, nombre) " +
                    "VALUES (" + codigo + ", '" + nombre + "')");
            }
            //Cerramos la base de datos
            db.close();
        }
    }
}
```

Ubicación de la base de datos

Para comprobar que todo funcionó correctamente y que los registros se han insertado debemos chequear lo siguiente. En primer lugar veamos dónde se ha creado nuestra base de datos. Todas las bases de datos SQLite creadas por aplicaciones Android se almacenan en la memoria del teléfono en un fichero con el mismo nombre de la base de datos situado en una ruta que sigue el siguiente patrón:

`/data/data/[paquete.java.de.la.aplicacion]/databases/nombre_base_datos`

Para comprobar esto podemos hacer lo siguiente. Una vez ejecutada por primera vez desde Eclipse la aplicación de ejemplo sobre el emulador de Android (y por supuesto antes de cerrarlo) podemos ir a la perspectiva “**DDMS**” (**Dalvik Debug Monitor Server**) de Eclipse y en la solapa “**File Explorer**” podremos acceder al sistema de archivos del emulador, donde podremos buscar la ruta indicada de la base de datos. Podemos ver esto en la siguiente imagen (click para ampliar):



Con esto ya comprobamos al menos que el archivo de nuestra base de datos se ha creado en la ruta correcta.

Comprobando tablas y datos

Para comprobar que tanto las tablas creadas como los datos insertados también se hayan incluido correctamente en la base de datos, podemos recurrir a dos posibles métodos.

- Transferir la base de datos a nuestro PC y consultarla con cualquier administrador de bases de datos SQLite.
- Acceder directamente a la consola de comandos del emulador de Android y utilizar los comandos existentes para acceder y consultar la base de datos SQLite.

En el primer método, el archivo de la base de datos podemos transferirlo a nuestro PC utilizando el botón de descarga situado en la esquina superior derecha del explorador de archivos (remarcado en rojo en la imagen anterior).

Junto a este botón aparecen otros dos para hacer la operación contraria (copiar un archivo local al sistema de archivos del emulador) y para eliminar archivos del emulador.

Una vez descargado el archivo a nuestro sistema local, podemos utilizar cualquier administrador de SQLite para abrir y consultar la base de datos, por ejemplo "SQLite Administrator" que es freeware (<http://sqliteadmin.orbmu2k.de/>).

En el segundo método en vez de descargar la base de datos a nuestro sistema local, somos nosotros los que accedemos de forma remota al emulador a través de su consola de comandos (*shell*).

Para ello, con el emulador de Android aún abierto, debemos abrir una consola de MS-DOS y utilizar la utilidad adb.exe (**Android Debug Bridge**) situada en la carpeta platform-tools del SDK de Android (en mi caso: `c:\android-sdk-windows\platform-tools\`).

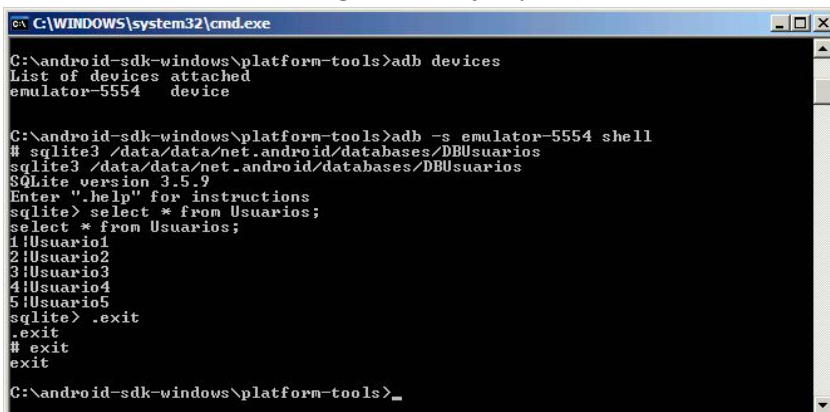
En primer lugar consultaremos los identificadores de todos los emuladores en ejecución mediante el comando "**adb devices**". Esto nos debe devolver una única instancia si sólo tenemos un emulador abierto, que en mi caso particular se llama "emulator-5554".

Tras conocer el identificador de nuestro emulador, vamos a acceder a su shell mediante el comando "**adb -s identificador-del-emulador shell**".

Una vez conectados, ya podemos acceder a nuestra base de datos utilizando el comando `sqlite3` pasándole la ruta del archivo:

"sqlite3 /data/data/[paquete.java.de.la.aplicacion]/databases/DBUsuarios".

Si todo ha ido bien, debe aparecernos el *prompt* de SQLite "`sqlite>`", lo que nos indicará que ya podemos escribir las consultas SQL necesarias sobre nuestra base de datos. Nosotros vamos a comprobar que existe la tabla Usuarios y que se han insertado los cinco registros de ejemplo. ("`SELECT * FROM Usuarios;`").



```
C:\WINDOWS\system32\cmd.exe
C:\android-sdk-windows\platform-tools>adb devices
List of devices attached
emulator-5554    device

C:\android-sdk-windows\platform-tools>adb -s emulator-5554 shell
# sqlite3 /data/data/net.android/databases/DBUsuarios
sqlite3 /data/data/net.android/databases/DBUsuarios
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> select * from Usuarios;
select * from Usuarios;
1|Usuario1
2|Usuario2
3|Usuario3
4|Usuario4
5|Usuario5
sqlite> .exit
.exit
# exit
exit
C:\android-sdk-windows\platform-tools>
```

Con esto ya hemos comprobado que nuestra base de datos se ha creado correctamente, que se han insertado todos los registros de ejemplo y que todo funciona según se espera.

En los siguientes artículos comentaremos las distintas posibilidades que tenemos a la hora de manipular los datos de la base de datos (insertar, eliminar y modificar datos) y cómo podemos realizar consultas sobre los mismos, ya que [como

siempre] tendremos varias opciones disponibles.