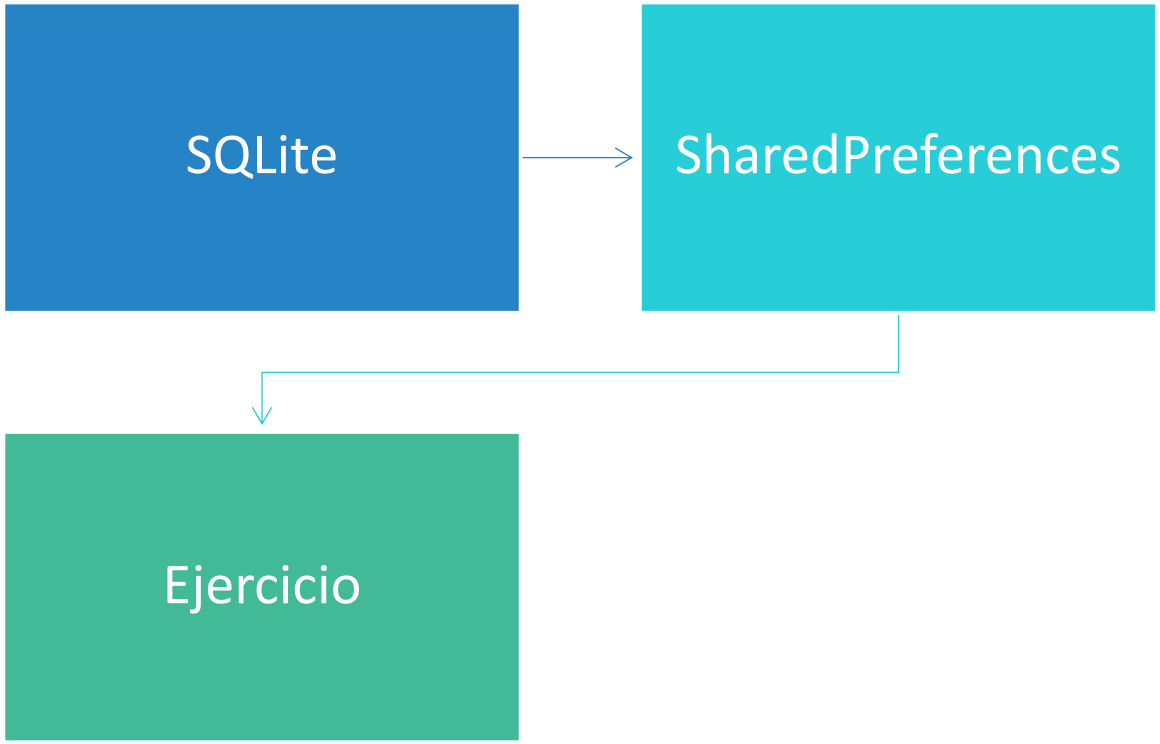
2º DAM

Programación Multimedia y Dispositivos Móviles

Programación Android

José A. Lara

Contenido



SQLite

SQLite

SQLite es una pequeña librería programada en lenguaje C que implementa un completo motor de base de datos multiplataforma que no precisa configuración. Se distribuye bajo licencia de dominio público. Es muy rápido y la ventaja fundamental es que permite utilizar un amplio subconjunto del lenguaje estándar SQL. SQLite destaca también por su versatilidad.

Combina el motor y el interfaz de la base de datos en una única biblioteca, y almacena los datos en un único archivo de texto plano.

Su potencia se basa fundamentalmente en la simplicidad, lo que hace que no sea una buena solución en entornos de tráfico muy elevado y/o alto acceso concurrente a datos. SQLite encapsula toda la base de datos en un único fichero.

La clase SQLiteOpenHelper

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 herede de ella.

La clase SQLiteOpenHelper tiene un constructor y dos métodos abstractos, onCreate() y onUpgrade(), que se deben personalizar con el código necesario para crear la base de datos de la aplicación y para actualizar su estructura respectivamente.

La clase SQLiteOpenHelper

```
class AlumnosContract {
    companion object {
        val VERSION = 1
        class Entrada : BaseColumns {
            companion object {
                val NOMBRE_TABLA = "alumnos"
                val COLUMNA_ID = "id"
                val COLUMNA_NOMBRE = "nombre"
            }
        }
    }
}
```

```
class DatabaseHelper(context: Context):
    SQLiteOpenHelper(context, AlumnosContract.Companion.Entrada.NOMBRE_TABLA, factory: null,
                                                    AlumnosContract.Companion.VERSION) {
```

Programación Android

La clase SQLiteOpenHelper

```
public class DBOpenHelper extends SQLiteOpenHelper {
    private static final String NOMBRE_BD = "usuarios";
    private static final int VERSION_BD = 1;
    private static DBOpenHelper dbOpen;

    private DBOpenHelper(@Nullable Context context) {
        super(context, NOMBRE_BD,  factory: null, VERSION_BD);
    }

    public static DBOpenHelper getInstance(Context context){
        if (dbOpen==null)
            dbOpen = new DBOpenHelper(context);
        return dbOpen;
    }
```

android

La clase SQLiteOpenHelper

A continuación se emplea el método onCreate() para crear una tabla usuario con los campos id, usuario, clave y nombre.

La sintaxis de SQLite es muy parecida a SQL y permite los tipos de datos habituales (varchar-nvarchar, text, integer, int, float, double, bit-boolean) así como todas las restricciones típicas de base de datos (PRIMARY KEY, FOREIGN KEY, …).

Este método (onCreate()) es invocado automáticamente por Android cuando detecta que la base de datos no existe, es decir, se ejecutará una única vez tras lanzar la aplicación por vez primera e intentar acceder a la base de datos.

La clase SQLiteOpenHelper

Para crear una tabla con clave autonumérica basta con que el tipo de la clave primaria sea integer, de forma que si al insertar no se establece un valor para dicho campo la base de datos lo genera de forma automática.

```
companion object{
    val CREATE_ALUMNOS_TABLA="CREATE TABLE "+ AlumnosContract.Companion.Entrada.NOMBRE_TABLA+
            " ("+AlumnosContract.Companion.Entrada.COLUMNA_ID+" TEXT PRIMARY KEY, "+
            AlumnosContract.Companion.Entrada.COLUMNA_NOMBRE+" TEXT)"

    val REMOVE_ALUMNOS_TABLA="DROP TABLE IF EXISTS "+AlumnosContract.Companion.Entrada.NOMBRE_TABLA
}

override fun onCreate(db: SQLiteDatabase?) {
    db?.execSQL(CREATE_ALUMNOS_TABLA)
}

override fun onUpgrade(db: SQLiteDatabase?, p1: Int, p2: Int) {
    db?.execSQL(REMOVE_ALUMNOS_TABLA)
    onCreate(db)
}
```

La clase SQLiteOpenHelper

Para crear una tabla con clave autonumérica basta con que el tipo de la clave primaria sea integer, de forma que si al insertar no se establece un valor para dicho campo la base de datos lo genera de forma automática.

```
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    try{
        sqLiteDatabase.execSQL("CREATE TABLE alumnos ("+
                "id TEXT PRIMARY KEY,"+
                "nombre TEXT)");
    }catch (Exception e){
        e.printStackTrace();
    }
}

@Override
public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
    sqLiteDatabase.execSQL("DROP TABLE IF EXISTS alumnos;");
    onCreate(sqLiteDatabase);
}
```

android

SQLite Database

Es la clase manejadora de una base de datos SQLite. Desde esta clase se podrán crear y borrar elementos de la base de datos y ejecutar comandos SQL sobre la misma.

A través de esta clase se pueden definir métodos que devuelvan valores o que realicen cualquier operación como insertar, actualizar…, en la base de datos.

El siguiente método devuelve los datos de usuario de una fila de la tabla dado un id. El resultado de una consulta se devuelve en forma de cursor, donde se podrá acceder a cada una de las columnas especificadas a través del índice numérico (comenzando en cero).

SQLite Database

Se pueden lanzar consultas en SQL puro a través del método **rawQuery** o utilizar el método **query**:

```
fun getAlumno(id:String):Alumno{
    var item:Alumno?=null
    //Abrimos la BD en modo lectura
    val db:SQLiteDatabase=helper?.readableDatabase!!
    //especificamos las columnas a leer
    val columnas=arrayOf(AlumnosContract.Companion.Entrada.COLUMNA_ID, AlumnosContract.Companion.Entrada.COLUMNA_NOMBRE)
    //creamos un cursor
    //val c: Cursor =db.query(AlumnosContract.Companion.Entrada.NOMBRE_TABLA,
    //   columnas," id=?",arrayOf(id),null,null,null)
    //creamos un cursor
    //Cursor c = db.query("alumnos", columnas, " id=?", id, null, null, null);
    val consulta = "SELECT * FROM alumnos WHERE id='" + id[0] + "';"
    val c = db.rawQuery(consulta,  selectionArgs: null)
    //recorremos el cursor
    while (c.moveToNext()){
        item=Alumno(c.getString(c.getColumnIndexOrThrow(AlumnosContract.Companion.Entrada.COLUMNA_ID)),
            c.getString(c.getColumnIndexOrThrow(AlumnosContract.Companion.Entrada.COLUMNA_NOMBRE)))
    }
    c.close()
    db.close()
    return item!!
}
```

SQLite Database

```
public Alumno getAlumno(Context context, String[] id) {
    Alumno item = null;
    //Abrimos la BD en modo lectura
    SQLiteDatabase db = DBOpenHelper.getInstance(context).getReadableDatabase();
    //especificamos las columnas a leer
    String[] columnas = {"id", "nombre"};
    //creamos un cursor
    //Cursor c = db.query("alumnos", columnas, " id=?", id, null, null, null);
    String consulta="SELECT * FROM alumnos WHERE id='"+id[0]+"';";
    Cursor c = db.rawQuery(consulta, selectionArgs: null);
    //recorremos el cursor
    while (c.moveToNext()) {
        item = new Alumno(c.getString(c.getColumnIndexOrThrow( columnName: "id")),
                c.getString(c.getColumnIndexOrThrow( columnName: "nombre")));
    }
    c.close();
    db.close();
    return item;
}
```

android

SQLite Database

El método query() permite ejecutar consultas sin necesidad de conocer la sintaxis SQL, estableciendo a través de sus argumentos los parámetros de configuración de la consulta:

- nombre de la tabla
- columnas a recuperar
- Condición
- parámetros de la condición (si la condición incluye ?)
- cláusula de agrupación (group by)
- cláusula de condición sobre la agrupación (having)
- cláusula de ordenación (order by)

Programación Android

SQLite Database

```
fun getAlumnos(): MutableList<Alumno>{
    val items:MutableList<Alumno> =mutableListOf()
    //Abrimos la BD en modo lectura
    val db:SQLiteDatabase=helper?.readableDatabase!!
    //especificamos las columnas a leer
    val columnas=arrayOf(AlumnosContract.Companion.Entrada.COLUMNA_ID, AlumnosContract.Companion.Entrada.COLUMNA_NOMBRE)
    //creamos un cursor
    val c: Cursor =db.query(AlumnosContract.Companion.Entrada.NOMBRE_TABLA,
                        columnas, selection: null, selectionArgs: null, groupBy: null, having: null, orderBy: null)
    //recorremos el cursor
    while (c.moveToNext()){
        items.add(Alumno(c.getString(c.getColumnIndexOrThrow(AlumnosContract.Companion.Entrada.COLUMNA_ID)),
            c.getString(c.getColumnIndexOrThrow(AlumnosContract.Companion.Entrada.COLUMNA_NOMBRE))))
    }
    c.close()
    db.close()
    return items
}
```

android

Programación Android

SQLite Database

```
public ArrayList<Alumno> getAlumnos(Context context) {
    ArrayList<Alumno> items = new ArrayList<~>();
    //Abrimos la BD en modo lectura
    SQLiteDatabase db = DBOpenHelper.getInstance(context).getReadableDatabase();
    //especificamos las columnas a leer
    String[] columnas = {"id", "nombre"};
    //creamos un cursor
    Cursor c = db.query( table: "alumnos", columnas, selection: null, selectionArgs: null,
            groupBy: null, having: null, orderBy: null);
    //recorremos el cursor
    while (c.moveToNext()) {
        items.add(new Alumno(c.getString(c.getColumnIndexOrThrow( columnName: "id")),
                c.getString(c.getColumnIndexOrThrow( columnName: "nombre"))));
    }
    c.close();
    db.close();
    return items;
}
```

android

SQLite Database

En el siguiente método se inserta una fila en la tabla usuario empleando también una consulta SQL. Cabe destacar que en este caso será imprescindible una copia en modo escritura de la base de datos.

```
fun newAlumno(item:Alumno){
    //Abrimos la BD en modo escritura
    val db:SQLiteDatabase=helper?.writableDatabase!!
    //mapeo de columnas con valores a insertar en la BD
    val values=ContentValues()
    values.put(AlumnosContract.Companion.Entrada.COLUMNA_ID,item.id)
    values.put(AlumnosContract.Companion.Entrada.COLUMNA_NOMBRE,item.nombre)
    //insertar una fila
    //db.insert(AlumnosContract.Companion.Entrada.NOMBRE_TABLA,null,values)
    val consulta = "INSERT INTO alumnos VALUES('" + item.id + "','"+item.nombre+"');"
    db.execSQL(consulta)
    db.close()
}
```

SQLite Database

En el siguiente método se inserta una fila en la tabla usuario empleando también una consulta SQL. Cabe destacar que en este caso será imprescindible una copia en modo escritura de la base de datos.

```
public void newAlumno(Context context, Alumno alumno) {
    //Abrimos la BD en modo escritura
    SQLiteDatabase db = DBOpenHelper.getInstance(context).getWritableDatabase();
    //mapeo de columnas con valores a insertar en la BD
    ContentValues values = new ContentValues();
    values.put("id", alumno.getId());
    values.put("nombre", alumno.getNombre());
    //insertar una fila
    //db.insert("alumnos", null, values);
    String sql="INSERT INTO alumnos VALUES('"+alumno.getId()+"','"+alumno.getNombre()+"');";
    db.execSQL(sql);
    db.close();
}
```

SQLite Database

Como hemos visto en los ejemplos, existen versiones sencillas para los métodos de acceso a la base de datos: insert, update, delete.

Para insert y update es necesario crear una colección ContentValues con los pares clave-valor que se van a insertar o actualizar respectivamente.

Para las consultas utilizaremos **rawQuery()** que nos devolverá un **Cursor**. Para el resto de operaciones utilizaremos **execSQL()**.

Programación Android

SQLite Database

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        var alumno=Alumno( id: "1", nombre: "Pepito Grillo")
        val alumnocrud=AlumnoCRUD( context: this)
        alumnocrud.newAlumno(alumno)
        alumno=alumnocrud.getAlumno( id: "1")
        val tvTexto=findViewById<TextView>(R.id.tvTexto)
        tvTexto.text=alumno.nombre
    }
}
```

Programación Android

SQLite Database

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Alumno alumno= new Alumno( id: "1", nombre: "Pepito Grillo");
        AlumnoCRUD alumnocrud=new AlumnoCRUD();
        alumnocrud.newAlumno( context: this, alumno);
        alumno=alumnocrud.getAlumno( context: this, id: "1");
        TextView tvTexto=findViewById(R.id.tvTexto);
        tvTexto.setText(alumno.getNombre());
    }
}
```

android

SharedPreferences

SharedPreferences

Para almacenar datos básicos Android proporciona el mecanismo de Shared Preferences, que permite almacenar, recuperar y compartir información entre aplicaciones a través de ficheros XML.

El acceso se permite entre otros en estos modos (que son los más usados):

- **MODE_PRIVATE** para la aplicación actual
- **MODE_APPEND** para añadir más preferencias

Programación Android

SharedPreferences

```
class MainActivity : AppCompatActivity(), View.OnClickListener {
    private var preferencias: SharedPreferences? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val btnGuardar = findViewById<Button>(R.id.btnGuardar)
        val btnRecuperar = findViewById<Button>(R.id.btnRecuperar)
        btnGuardar?.setOnClickListener(this)
        btnRecuperar?.setOnClickListener(this)
        preferencias = getSharedPreferences(getString(R.string.app_name), MODE_PRIVATE)
    }
```

```
    override fun onClick(view: View) {
        when (view.id) {
            R.id.btnGuardar -> {
                val editor = preferencias!!.edit()
                val etUsuario:EditText=findViewById(R.id.etUsuario)
                editor.putString("usuario", etUsuario.text.toString())
                editor.apply()
                val toast = Toast.makeText(
                    applicationContext,
                    text: "Usuario guardado correctamente",
                    Toast.LENGTH_SHORT
                )
                toast.show()
            }
            R.id.btnRecuperar -> {
                val usuario = preferencias!!.getString("usuario", null)
                val tvUsuario:TextView = findViewById(R.id.tvUsuario)
                tvUsuario.text = usuario
                val toast2 = Toast.makeText(
                    applicationContext,
                    text: "Usuario recuperado correctamente",
                    Toast.LENGTH_SHORT
                )
                toast2.show()
            }
        }
    }
}
```

Programación Android

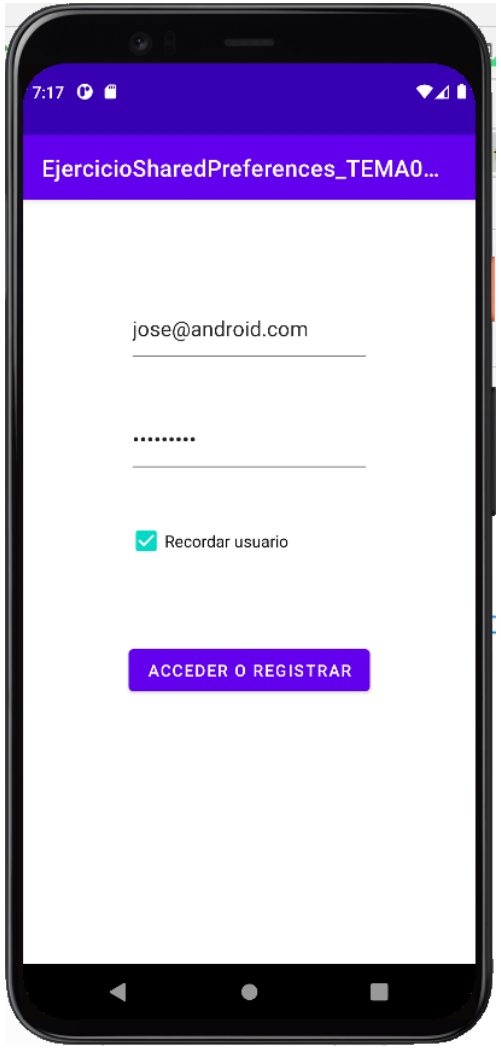
SharedPreferences

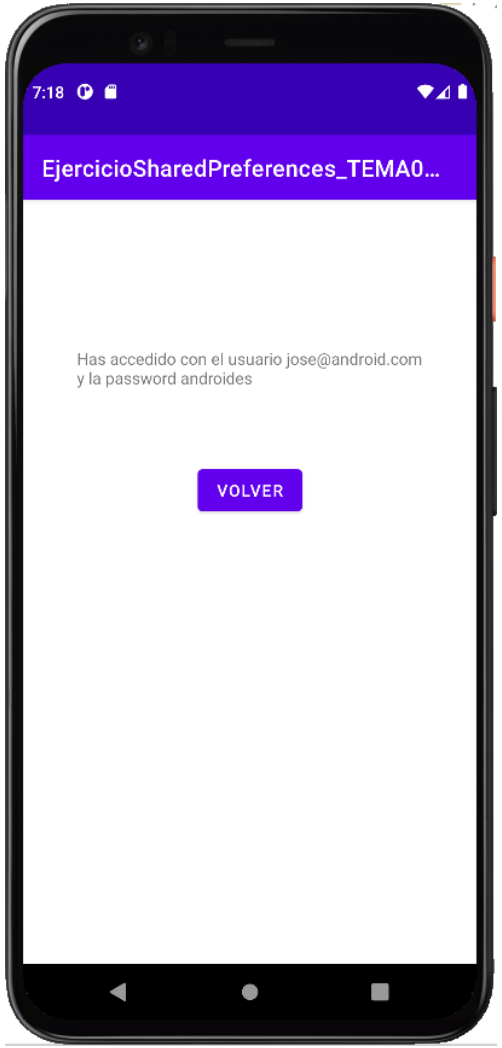
```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{
    private SharedPreferences preferencias;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnGuardar= findViewById(R.id.btnGuardar);
        Button btnRecuperar=findViewById(R.id.btnRecuperar);
        btnGuardar.setOnClickListener(this);
        btnRecuperar.setOnClickListener(this);
        preferencias=getSharedPreferences(getString(R.string.app_name), Context.MODE_PRIVATE);
    }
    @Override
    public void onClick(View view) {
        switch (view.getId()){
            case R.id.btnGuardar:
                SharedPreferences.Editor editor = preferencias.edit();
                EditText etUsuario=findViewById(R.id.etUsuario);
                editor.putString( s: "usuario",etUsuario.getText().toString());
                editor.apply();
                Toast toast=Toast.makeText(getApplicationContext(),
                        text: "Usuario guardado correctamente",Toast.LENGTH_SHORT);
                toast.show();
                break;
            case R.id.btnRecuperar:
                String usuario=preferencias.getString( s: "usuario", s1: null);
                TextView tvUsuario=findViewById(R.id.tvUsuario);
                tvUsuario.setText(usuario);
                Toast toast2=Toast.makeText(getApplicationContext(),
                        text: "Usuario recuperado correctamente",Toast.LENGTH_SHORT);
                toast2.show();
        }
    }
}
```

android

Programación Android

SharedPreferences





android

Ejercicio

Programación Android

Ejercicio

Crear una aplicación que al ejecutarse la primera vez genere una base de datos con una tabla que contenga las comunidades autónomas de España (id, nombre, imagen). La página de inicio es la lista cargada con las comunidades que aparecen en la base de datos, mostrando el campo nombre y la imagen.

Al pulsar sobre una de ellas nos mostrará un diálogo en el que se muestre el mensaje “Yo soy de…”

Al seleccionar una de las comunidades, se mostrará un formulario donde se podrá editar el nombre de la misma, realizando la actualización correspondiente de la Base de datos al pulsar un botón Guardar y mostrando de nuevo la lista actualizada.

También tendremos un menú con el que limpiar el listview (borrarlo) y recargarlo de nuevo con los valores almacenados en la base de datos.

android

Programación Android

Ejercicio

android