Persistencia en Android: ficheros y SQLite - Ejercicios

Índice

1. Uso de ficheros

En este ejercicio vamos a crear una aplicación que muestre un listado de cadenas por pantalla. Estas cadenas se almacenarán en un fichero de texto privado para la aplicación. Podremos añadir nuevas cadenas a partir de un cuadro de edición presente en la interfaz. Partiremos del proyecto *Ficheros* proporcionado en las plantillas de la aplicación. Debes seguir los siguientes pasos:

- Añadir un manejador al botón de la actividad principal para que cada vez que sea pulsado se guarde en un fichero de texto. El fichero se llamará `fichero.txt`. Al abrir el fichero para escritura se utilizará el parámetro `Context.MODE_APPEND`, con lo que cada nueva cadena se añadirá al final del fichero en el caso en el que éste ya existiera. Para escribir en el fichero utilizaremos el método `writeBytes` del objeto `DataOutputStream` correspondiente.
- Al pulsar el botón también se deberá borrar el contenido del elemento `EditText` (le asignamos a la vista una cadena vacía con el método `setText`).
- Ejecutamos la aplicación e introducimos algunas líneas en el fichero. Vamos a comprobar ahora que todo ha funcionado correctamente. Para ello accedemos al sistema de ficheros de nuestro dispositivo ejecutando el comando `./adb shell` dentro de la carpeta *platform-tools* de nuestra carpeta de instalación del SDK de Android.
- Comprueba que se ha creado el fichero *fichero.txt* en la carpeta */data/data/es.ua.jtech.android.ficheros/files/*. Examina su contenido ejecutando `cat fichero.txt`. Si algo no ha salido bien siempre podrás eliminar el fichero con `rm fichero.txt`.
- En la interfaz de la actividad se ha incluido una vista de tipo `TextView` debajo del botón. Añade un manejador para dicha vista, de tal forma que cada vez que se haga click sobre ella se muestre el contenido del fichero *fichero.txt*. Para ello leeremos el fichero línea a línea, añadiendo cada una al `TextView` por medio del método `append`.

Interfaz de la aplicación Ficheros

2. Persistencia con ficheros (*)

Seguramente habrás observado que si abandonas la aplicación anterior (pulsando el botón *BACK* del dispositivo), al volver a ejecutarla el contenido del `TextView` ha desaparecido. Debes volver a pulsar sobre la vista para que se vuelva a mostrar el contenido del fichero. Haz las modificaciones pertinentes para que esto no sea así; es decir, para que cuando vuelva a mostrarse la actividad tras haber sido eliminada de la memoria se muestre el `TextView` tal cual se veía antes de abandonar la aplicación.

Aviso:

En este ejercicio no se está pidiendo que cargues el contenido del fichero en el `TextView` nada más arrancar la aplicación. Puede darse el caso de que lo que esté mostrando el `TextView` cuando la actividad sea eliminada de memoria no se corresponda con el contenido actualizado del fichero.

Nota:

Para poder completar el ejercicio deberás repasar los manejadores de evento de la primera sesión del módulo de Android relacionados con el ciclo de vida de ejecución de actividades.

3. Base de datos: SQLiteOpenHelper

En las plantillas de la aplicación se incluye la aplicación *BaseDatos*. En el proyecto de la aplicación se incluye el esqueleto de una clase `MiAdaptadorBD` que tendremos que

completar. Se trata de un patrón que nos va a permitir acceder a una base de datos de usuarios dentro de la aplicación sin necesidad de hacer uso de código SQL.

La clase `MiAdaptadorBD` incluye a su vez otro patrón, en este caso implementado como una subclase de `SQLiteOpenHelper`. Éste nos obliga a definir qué ocurre cuando la base de datos todavía no existe y debe ser creada, y qué ocurre si ya existe pero debe ser actualizada porque ha cambiado de versión. Así el `SQLiteOpenHelper` que implementemos, en este caso MiOpenHelper, nos devolverá siempre una base de datos separándonos de la lógica encargada de comprobar si la base de datos existe o no.

En este primer ejercicio se pide hacer lo siguiente:

- Ejecutar la sentencia de creación de bases de datos (la tenemos declarada como constante de la clase) en el método `MiOpenHelper.onCreate`.
- Implementar también el método `onUpgrade`. Idealmente éste debería portar las tablas de la versión antigua a la versión nueva, copiando todos los datos. Nosotros vamos a eliminar directamente la tabla que tenemos con la sentencia SQL `"DROP TABLE IF EXISTS " + NOMBRE_TABLA` y volveremos a crearla.
- En el constructor de `MiAdaptadorBD` debemos obtener en el campo db la base de datos, utilizando `MiOpenHelper`.

4. Base de datos: inserción y borrado

Continuamos trabajando con el proyecto anterior. En este ejercicio completaremos el código relacionad con las sentencias de inserción y borrado. Para realizar la inserción vamos a hacer uso de un mecanismo que no hemos visto en la sesión de teoría, y que consiste en utilizar una sentencia de inserción SQL precompilada que se completará con los valores concretos a insertar en la base de datos justo antes de realizar dicha inserción. La ventaja de las sentencias compiladas es que evitan que se produzcan ataques de inyección de código.

Como se puede observar, se ha incluido un atributo a la clase que representa la inserción SQL:

```
private static final String INSERT = "insert into " + NOMBRE_TABLA +
        "("+COLUMNAS[1]+") values (?)";
```

En esta sentencia no se indica ningún valor concreto para la columna *nombre*. En el lugar en el que deberían aparecer los valores de dicho campo se ha escrito simplemente un símbolo de interrogación. También se ha añadido una instancia de la clase `SQLiteStatement` como parte de los atributos de la clase:

```
private SQLiteStatement insertStatement;
```

Realizamos los siguientes pasos:

- En el constructor de la clase `MiAdaptadorBD` llevamos a cabo la compilación de la sentencia:

```
this.insertStatement = this.db.compileStatement(INSERT);
```

- Una vez hecho esto, cada vez que deseemos insertar un nuevo usuario en la base de datos, deberemos dar valores concretos a la columna *nombre*, y ejecutar la sentencia. Para ello debemos añadir el siguiente código dentro del método `insert` de la clase `MiAdaptadorBD`:

```
this.insertStatement.bindString(1, nombreUsuario);
return this.insertStatement.executeInsert();
```

- Por último completamos el código del método `deleteAll`, cuyo contenido es eliminar a todos los usuarios de la base de datos. Para que además se devuelva el número de filas afectadas podemos insertar la siguiente línea en dicho método:

```
return db.delete(NOMBRE_TABLA, null, null);
```

Una vez hechos todos estos cambios podremos utilizar la clase `MiAdaptadorBD` para hacer operaciones con la base de datos de usuarios de manera transparente.

5. Base de datos: probar nuestro adaptador

Añade código en la actividad `Main` para eliminar todos los usuarios de la base de datos, añadir dos cualesquiera, y listarlos por medio de la vista de tipo `TextView` que se encuentra en el layout de dicha actividad.

Podemos comprobar mediante la línea de comandos (comando `./adb shell`) que la base de datos ha sido creada. Para ello puede ser útil hacer uso de los siguientes comandos:

```
#cd /data/data/es.ua.jtech.android.basedatos/databases
#sqlite3 misusuarios.db
sqlite> .schema
sqlite> .tables
sqlite> select * from usuarios;
```

6. Base de datos: cambios en la base de datos (*)

Ahora vamos a cambiar en la clase `MiAdaptadorBD` el nombre de la segunda columna, que en lugar de *nombre* se va a llamar *nombres*. Ejecutamos la aplicación y comprobamos que sale con una excepción. Comprueba por medio de LogCat cuál ha sido el error. ¿Cómo lo podemos solucionar?

Nota:

Pista: conforme hemos programado la clase `MiAdaptadorBD` y siguiendo el patrón de diseño de `SQLiteOpenHelper`, es posible arreglar el problema simplemente modificando el valor de un campo.