Aplicación Android de Asistencia al Caminante

Enrique R. Delgado Garrido

Almacenamiento de datos en Android.

Documento: Teórico-Práctico

Índice de contenido

A2.- Almacenamiento de datos en Android.

Existen cuatro maneras de almacenar datos en Android, a saber:

A.2.1 - Preferences – es la forma 'ligera' de almacenar y recuperar datos en Android. Utiliza el sistema de pares clave – valor con tipos de valor primitivos. Aunque sea trasparente para el programador es interesante saber que estos pares clave-valor se almacenan en formato xml. XXXX poner dirección web

En nuestro caso el código utilizado par mantener entre sesiones el nombre del caminante en la actividad PanatallaCAMINANTE de la aplicación Caminante_DATOS es:

```
import android.content.SharedPreferences;
```

…

```
/** variables para el almacenamiento del Nombre_del_caminante
 *  en -preferences- **/
private static final    String      DEFAULT_WALKER    = "Nombre";
private                       String      wALKER            = DEFAULT_WALKER;
private static final    String      KEY_WALKER  = "NOMBRE_CAMINANTE";
private static final    String      PREFS             = "caminante.prefs";
```

…

```
public void onCreate(Bundle savedInstanceState) {
      ...
      // mantener el nombre del caminante entre sesiones
      SharedPreferences settings = getSharedPreferences(PREFS, 0);
      wALKER  = settings.getString(KEY_WALKER,  DEFAULT_WALKER);
      ...
}
```

…

```
public void onClick(View vista) {
      // captura del nombre del caminante
      wALKER = ((EditText)findViewById(R.id.editTextName)).getText().toString();
      // almacenamiento del nombre con preferences
      SharedPreferences settings = getSharedPreferences(PREFS, 0);
      SharedPreferences.Editor editor = settings.edit();
      editor.putString(KEY_WALKER, wALKER);
      editor.commit();
      ...
}
```

A.2.2.- Archivos – Esta posibilidad se puede ejercer guardando los ficheros en la memoria del dispositivo (almacenamiento interno) o en un medio de almacenamiento externo como una Sdcar que es lo que hacemos en el programa Caminante_DATOS como se muestra a continuación.

En la Activity PantallaCAMINANTE

```
/** variable objeto de la clase FicherosCAMINANTE
 *  para lo referente a los fcheros de almacenamiento **/
private FicherosCAMINANTE          dataFiles;
```

```
//@Override
public void onClick(View vista) {
      ...
      // Filtro de los controles
      // Botón de inicio de la captura de datos
      if(vista.getId()==R.id.buttonStart) {
            ...
            //paso del nombre del caminante para su uso en la clase
            // FicherosCAMINANTE - uso de un constructor -ficticio-
            dataFiles = new FicherosCAMINANTE(wALKER, 0);
            // Llamada al constructor -real- de FicherosCAMINANTE
            dataFiles = new FicherosCAMINANTE(this);
            ...
      }
      // Botón de terminación el proceso de captura de datos
      if(vista.getId()==R.id.buttonFinish) {
            ...
            stopProcess();
            ...
      }
      ...
}

protected void stopProcess() {
      ...
      // cierra los ficheros (y lanza el correo)
      dataFiles.fileClose(this);
}

protected void onDestroy() {
      ...
            stopProcess();
      ...
}
```

```
private SensorEventListener sensorEventListener = new SensorEventListener() {
      ...
      public void onSensorChanged(SensorEvent event) {
            ...
                  //escritura los dato en el fichero del acelerómetro
                  // por medio del objeto de la clase FicherosCAMINANTE
                  dataFiles.addDatoACEL(coordinatesAccelerometer[0],
```

```
                                                coordinatesAccelerometer[1],
                                                coordinatesAccelerometer[2]);
            ...
      }
      ...
}
```

En la clase FicherosCAMINANTE

```
import java.io.BufferedWriter;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;

public class FicherosCAMINANTE {
/** Variables para el uso del nombre del caminante **/
      static      String                  nameDirectory     = "Caminar_";
      static      String                  walker                  = "";

/** Variables para el trataminto de los ficheros y su escritura **/
      private     BufferedWriter          bufferedWriterACEL;
      private File                        txtFileACEL             =     null;
      private     BufferedWriter          bufferedWriterORIE;
      private File                        txtFileORIE             =     null;

      private String                headerLine;
      private String                dataLine;

      private String    nameFileACEL = "ACEL_"+timeStamp.toString()+".txt";
      private String    nameFileORIE = "ORIE_"+timeStamp.toString()+".txt";
...
/** Variables para la localización de los ficheros y su nombre **/
      private static final    String EXT_PATH =
Environment.getExternalStorageDirectory()+"/Android/data/com.caminante.datos/";
      private static          String EXT_PATH_walker = null;

/** Constructor -ficticio- para paso del nombre del caminante
 *  se aprovecha para la creación del nombre del directorio de almacenamiento
 *  de los ficheros. Puede obviarse sustituyéndolo por otras opciones
 *  para el paso del nombre
 */

      public FicherosCAMINANTE(String wALKER, Integer nUM) {
            nameDirectory = "Caminar_" + wALKER;
            walker = wALKER;
      }
/** Constructor -real-
 */
```

```
      public FicherosCAMINANTE(Context ctx){

            EXT_PATH_walker = EXT_PATH + nameDirectory;
            try {
                  if
(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
                        // creación del directorio raiz si no existe
                        new File(EXT_PATH).mkdirs();
                        // creación del directorio para el nombre del caminante
                        new File(EXT_PATH_walker).mkdirs();
                  } else {
                        File dir = new File(EXT_PATH_walker);
                        if (!dir.exists()) {
                              // creación del directorio raiz porqué no existe
                              new File(EXT_PATH_walker).mkdirs();
                        }
                  }
                  //llamada que crea los ficheros de texto si no existen
                  FicheroACELEROMETRO(ctx);
                  ...
            } catch (Exception e) {
                  Log.w("ExternalStorage", "ErrorDIRECTORY" + e.getMessage());
            }
      }
/**
 * @param ctx
 */
      protected void FicheroACELEROMETRO(Context ctx) {

            try {
                  // crea el fichero si no existe y lo asigna si ya existe
                  txtFileACEL = new File(EXT_PATH_walker, nameFileACEL);
                  if (!txtFileACEL.exists()) {
                        txtFileACEL.createNewFile();
                  } else {
                        txtFileACEL = ctx.getFileStreamPath(nameFileACEL);
                  }
                  // escibe la linea de cabecera
                  bufferedWriterACEL = new BufferedWriter(new
                              OutputStreamWriter( new
FileOutputStream(txtFileACEL)));
                  headerLine = String.format("%s\t%s\t%s\t%s\n",

"coorAcel_0","coorAcel_1","coorAcel_2",
                                          "tiempo");
                  bufferedWriterACEL.write(headerLine);
                  bufferedWriterACEL.flush();
            } catch (IOException ioe) {
                  Log.w("ExternalStorage", "Error writing" + nameFileACEL, ioe);
            } finally {
```

```
            try {
                  if (bufferedWriterACEL == null) {
                        bufferedWriterACEL.close();
                  }
            } catch (Exception e) {
                  Log.d("ExceptionACEL: ", "finally: " + e.getMessage());
            }
      }
      }

/** Módulo que escribe los datos del acelerómetro en su fichero
 * @param ca0
 * @param ca1
 * @param ca2
 */
      public void addDatoACEL(float ca0, float ca1, float ca2){
            try{
                  ...
                  // dar formato a la linea de datos y escribirlos
                  dataLine = String.format("%.10f\t%.10f\t%.10f\t%d\n",
                                                            ca0,     ca1,     ca2,
timeStamp);
                  bufferedWriterACEL.write(dataLine);
                  bufferedWriterACEL.flush();
            } catch (Exception e) {
                  Log.d("Exception: ", "escribirACEL: " + e.getMessage());
            } finally {
            try {
                  if (bufferedWriterACEL == null) {
                        bufferedWriterACEL.close();
                  }
            } catch (Exception e) {
                  Log.d("Exception: ", "finallyACEL: " + e.getMessage());
            }
      }
      }
/** Módulo que escribe los datos de orientación en su fichero
 */
      public void addDatoORIE(float cm0, float cm1, float cm2){
            ...
      }
/** módulo para cerrar los ficheros ...
 * @param ctx
 */
      public void fileClose(Context ctx){

            try {
                  bufferedWriterACEL.close();
            } catch (IOException ioe) {
            ...
```

```
            } finally {
            try {
                  if (bufferedWriterACEL == null) {
                        bufferedWriterACEL.close();
            }
            } catch (IOException ioe) {
                  Log.d("IOException: ", "finally ACEL: " + ioe.getMessage());
                  ...
            }
      }
      ...

}
```

Mirar BufferReader http://developer.android.com/reference/java/io/BufferedReader.html

A.2.3.- Bases de datos. Como ya citamos Android soporta un API para SQLite. Todas las bases de datos SQLite y otras, se guardan en /data/data/nombre_paquete/Base_de_datos y es accedida en pricipio solo por la propia aplicación.

La primera versión del programa Caminante_DATOS utilizaba este método de almacenamiento, el código era el siguiente, teniendo en cuenta que estábamos utilizando un servicio.

```
public class ServicioCAMINANTE extends Service {
      ...
      private BDatosCAMINANTE      bDatos;
      ...
      public void onStart(Intent intent, int startId){
                  ...
                  bDatos = new BDatosCAMINANTE(CAMINANTE, 0);
                  bDatos = new BDatosCAMINANTE(this);
                  ...
      }

      public void onDestroy(){
            super.onDestroy();
            this.finalizarServicioEvento();
      }

      private void inicializarServicioEvento() {
            intent.setClass(this,
                  Sensores.class).setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            Bundle bundle = new Bundle();
            bundle.putString("CAMINANTE", CAMINANTE);
            intent.putExtras(bundle);
            startActivity(intent); **/
```

```
            this.startActivity(new Intent(getApplicationContext(),
            SensoresCAMINANTE.class).setFlags(Intent.FLAG_ACTIVITY_NEW_TASK));
            ...
      }

      private void finalizarServicioEvento() {
                  ...
                  bDatos.closeBDatos();
                  ...
      }

}
```

```
public class SensoresCAMINANTE extends Activity implements SensorEventListener
{     ...
      private BDatosCAMINANTE bDatos;
      private float[]         coordenadasAceleracion  = new float[3];
      private float[]         coordenadasOrientacion  = new float[3];
      ...
      @Override
      public void onCreate(Bundle savedInstanceState) {
            ...
            bDatos = mySingleton.getBDatos();
            ...
      }
      ...
      public void onSensorChanged(SensorEvent event) {
            //implantación de los escuchadores
                  ...
                  bDatos.addDato(coordenadasAceleracion[0],
                                    coordenadasAceleracion[1],
                                    coordenadasAceleracion[2],
                                    coordenadasOrientacion[0],
                                    coordenadasOrientacion[1],
                                    coordenadasOrientacion[2]);
                  ...
            }
      }
```

```
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabaseCorruptException;

public class BDatosCAMINANTE { //extends SensoresCAMINANTE{

      static      SQLiteDatabase    bdSQLite;
```

```
      static      String                  nombreBDatos;
      ...
      private     String CREATE_TABLE =
            "CREATE TABLE IF NOT EXISTS " + nombreTabla +
            " (" + "ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
            "coorAcel_0 float, " + "coorAcel_1 float, " + "coorAcel_2 float, " +
            "coorMagn_0 float, " + "coorMagn_1 float, " + "coorMagn_2 float, " +
            "tiempo long);";

      public BDatosCAMINANTE(String CAMINANTE, Integer NUM) {
            nombreBDatos = "Caminar_" + CAMINANTE;
      }
      private static final String EXT_PATH =
            Environment.getExternalStorageDirectory()
            +"/Android/data/com.caminante.datos_eventos_0_x/";

      public BDatosCAMINANTE(Context ctx){

            File dbFile=null;
            if
(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
                  new File(EXT_PATH).mkdirs();
                  dbFile = new File(EXT_PATH, nombreBDatos);
            } else {
                  dbFile = ctx.getDatabasePath(nombreBDatos);
            }

            bdSQLite = SQLiteDatabase.openOrCreateDatabase(dbFile, null);
            //creación de tablas
            bdSQLite.execSQL(CREATE_TABLE);
            }

      public long addDato(float ca0, float ca1, float ca2, float cm0, float cm1,
float cm2){
            ...
            Long timeStamp =
 new Timestamp(Calendar.getInstance().getTimeInMillis()).getTime();
            ContentValues cV = new ContentValues();
            cV.put("coorAcel_0", ca0);
            cV.put("coorAcel_1", ca1);
            cV.put("coorAcel_2", ca2);
            cV.put("coorMagn_0", cm0);
            cV.put("coorMagn_1", cm1);
            cV.put("coorMagn_2", cm2);
            cV.put("tiempo", timeStamp);
            return bdSQLite.insert(nombreTabla, null, cV);
            ...
      }

      public void closeBDatos(){
```

```
            bdSQLite.close();
      }
}
```

A.2.4.- Red. Por último diremos que Android también permite el almacenamiento de datos en la red sean estos base de datos u otro tipo de ficheros, para lo cual están a nuestra disposición los paquetes java.net.* y android.net:*