Manejo de Ficheros en Memoria Externa en Android con Java

1. Introducción a la Memoria Externa

La **Memoria Externa (External Storage)** en Android se refiere al almacenamiento que no es privado de la aplicación. Puede ser:

- **Almacenamiento Compartido/Principal:** Una partición del almacenamiento interno del dispositivo a la que otras aplicaciones pueden acceder.
- **Tarjeta SD Removible:** Una tarjeta SD física insertada por el usuario.

Características clave:

- **Público y Compartido:** Los archivos aquí pueden ser leídos y modificados por otras aplicaciones, y por el usuario directamente (conectando el dispositivo a un PC).
- **Persistencia:** Los archivos **no se eliminan** automáticamente al desinstalar la aplicación.
- **Permisos: Requiere permisos de usuario** para leer y escribir.
- **Disponibilidad:** El almacenamiento externo puede no estar siempre disponible (ej. si la tarjeta SD se extrae o se monta en un PC).

---

2. Tipos de Archivos en Memoria Externa

Existen dos tipos principales de directorios para tu aplicación en la memoria externa:

A. Archivos Específicos de la Aplicación (App-Specific Files)

- Almacenados en un subdirectorio privado de tu aplicación en el almacenamiento externo (ej. /Android/data/com.tudominio.tuapp/files).
- **Ventaja:** Se eliminan cuando la aplicación se desinstala.
- **Ventaja:** No son visibles en la Galería de medios (si se almacenan en el directorio files/).
- **Acceso:** Se obtienen con Context.getExternalFilesDir().

B. Archivos Compartidos/Públicos (Shared/Public Files)

- Almacenados en directorios públicos como Pictures/, DCIM/, Downloads/.
- **Ventaja:** Accesibles para todas las aplicaciones y para el usuario.
- **Desventaja:** No se eliminan al desinstalar la aplicación.
- **Acceso:** Se obtienen con Environment.getExternalStoragePublicDirectory() (obsoleto, ahora usar MediaStore o rutas estáticas como Environment.DIRECTORY_PICTURES).

---

3. Permisos Necesarios (¡Crucial!)

Para leer y escribir en el almacenamiento externo, debes declarar los permisos en el AndroidManifest.xml:

XML

```
<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <application ...>
        ...
    </application>
</manifest>
```

Permisos en Tiempo de Ejecución (Runtime Permissions)

A partir de Android 6.0 (API 23), no basta con declararlos; **debes solicitarlos al usuario en tiempo de ejecución**.

Java

```
// MainActivity.java
```

```
private static final int PERMISSION_REQUEST_CODE = 100;

private void requestStoragePermission() {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {
        // Debemos solicitar el permiso
        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                PERMISSION_REQUEST_CODE);
    } else {
        // El permiso ya fue otorgado, puedes proceder
        Log.d("ExternalStorage", "Permiso de escritura ya otorgado.");
        // Aquí podrías llamar a tu función para escribir/leer
        // guardarArchivoEnExternalStorage("Hola desde externo!");
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == PERMISSION_REQUEST_CODE) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            // Permiso concedido
            Log.d("ExternalStorage", "Permiso concedido. Puedes escribir/leer.");
            // guardarArchivoEnExternalStorage("Hola desde externo!");
        } else {
            // Permiso denegado
            Toast.makeText(this, "Permiso de almacenamiento denegado.", Toast.LENGTH_SHORT).show();
        }
    }
}
```

---

4. Comprobar la Disponibilidad del Almacenamiento Externo

Siempre debes verificar si el almacenamiento externo está montado y disponible para lectura/escritura antes de intentar acceder a él.

Java

```
public static boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    return Environment.MEDIA_MOUNTED.equals(state);
}

public static boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    return Environment.MEDIA_MOUNTED.equals(state) ||
            Environment.MEDIA_MOUNTED_READ_ONLY.equals(state);
}
```

---

5. Escribir (Guardar) un Archivo en Memoria Externa

Usaremos getExternalFilesDir() para un archivo específico de la aplicación.

Java

```
// MainActivity.java
private static final String FILE_NAME_EXTERNAL = "my_external_data.txt";

public void guardarArchivoEnExternalStorage(String data) {
    if (!isExternalStorageWritable()) {
```

```
        Toast.makeText(this, "Almacenamiento externo no disponible para escritura.", Toast.LENGTH_SHORT).show();
        return;
    }

    // Obtiene el directorio de archivos específico de tu app en el almacenamiento externo.
    // Usar null como tipo retorna el directorio raíz de la app-specific files.
    // También puedes usar Environment.DIRECTORY_DOCUMENTS, etc., para subdirectorios organizados.
    File directory = getExternalFilesDir(null); // Esto crea la ruta: /Android/data/com.tudominio.tuapp/files

    if (directory == null) {
        Toast.makeText(this, "No se pudo obtener el directorio externo.", Toast.LENGTH_SHORT).show();
        return;
    }

    File file = new File(directory, FILE_NAME_EXTERNAL);
    FileOutputStream fos = null;

    try {
        fos = new FileOutputStream(file); // Abre el flujo de salida al archivo
        fos.write(data.getBytes());
        Log.d("ExternalStorage", "Archivo guardado en: " + file.getAbsolutePath());
        Toast.makeText(this, "Archivo externo guardado!", Toast.LENGTH_SHORT).show();

    } catch (IOException e) {
        e.printStackTrace();
        Toast.makeText(this, "Error al guardar archivo externo.", Toast.LENGTH_SHORT).show();
    } finally {
        if (fos != null) {
            try {
                fos.close();
            } catch (IOException e) {
```

```
            e.printStackTrace();
        }
    }
  }
}
```

---

6. Leer (Recuperar) un Archivo de Memoria Externa

Para leer el archivo que acabamos de guardar:

Java

```
// MainActivity.java
public String leerArchivoDeExternalStorage() {
  if (!isExternalStorageReadable()) {
    Toast.makeText(this, "Almacenamiento externo no disponible para lectura.", Toast.LENGTH_SHORT).show();
    return "ERROR: Almacenamiento externo no disponible";
  }

  File directory = getExternalFilesDir(null); // Mismo directorio usado para guardar
  if (directory == null) {
    return "ERROR: No se pudo obtener el directorio externo.";
  }

  File file = new File(directory, FILE_NAME_EXTERNAL);
  if (!file.exists()) {
    return "ERROR: Archivo no encontrado en almacenamiento externo.";
  }

  FileInputStream fis = null;
```

```
    InputStreamReader isr = null;
    BufferedReader br = null;
    StringBuilder sb = new StringBuilder();

    try {
        fis = new FileInputStream(file);
        isr = new InputStreamReader(fis);
        br = new BufferedReader(isr);
        String text;

        while ((text = br.readLine()) != null) {
            sb.append(text).append("\n");
        }
        return sb.toString();

    } catch (IOException e) {
        e.printStackTrace();
        return "ERROR: Error de lectura de archivo externo.";
    } finally {
        try {
            if (br != null) br.close();
            if (isr != null) isr.close();
            if (fis != null) fis.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

---

7. Consideraciones Importantes y Migración a Scoped Storage (Android 10+)

Con Android 10 (API 29) se introdujo **Scoped Storage**, un cambio significativo en cómo las aplicaciones acceden al almacenamiento externo.

- **Objetivo:** Mejorar la privacidad del usuario y reducir la "contaminación" de los directorios públicos.
- **Funcionamiento:** Las aplicaciones tienen acceso total a sus propios directorios específicos de la aplicación (vistos arriba), pero el acceso a directorios públicos (DCIM, Pictures, Downloads) es más restringido.
- **MediaStore API:** Para interactuar con archivos de medios (imágenes, videos, audio) en directorios públicos, ahora se recomienda usar la **MediaStore API**.
- **requestLegacyExternalStorage:** Puedes optar temporalmente por el comportamiento antiguo añadiendo android:requestLegacyExternalStorage="true" a tu AndroidManifest.xml (solo hasta Android 10). **Esto es una solución temporal y no una buena práctica a largo plazo.**

  💡 **Recomendación:** Para nuevas aplicaciones o si tu targetSdkVersion es 29 o superior, prioriza getExternalFilesDir() para datos propios de la app y MediaStore API para compartir contenido multimedia. Evita el uso directo de Environment.getExternalStoragePublicDirectory() si puedes.

---

8. Resumen y Mejores Prácticas

- **Permisos:** ¡Siempre gestiona los permisos en tiempo de ejecución para WRITE_EXTERNAL_STORAGE y READ_EXTERNAL_STORAGE!
- **Disponibilidad:** Verifica isExternalStorageWritable() e isExternalStorageReadable() antes de operar.
- **Uso:**
  - **Memoria Interna:** Para datos privados y sensibles.
  - **Memoria Externa (App-Specific):** Para archivos grandes o no sensibles que no quieres que se eliminen si el usuario limpia la caché, pero sí si desinstala la app.
  - **Memoria Externa (Pública/Compartida) + MediaStore:** Para archivos que deben ser accesibles para otras apps o el usuario (fotos, documentos descargados).
- **Cierre de Flujos y Hilos:** Las mismas reglas se aplican aquí. Siempre cierra tus flujos y realiza operaciones de I/O en hilos secundarios.