DESARROLLO DE APLICACIONES EN ANDROID

SAMSUNG

Abraham Gutiérrez Rodríguez

Samsung TECH INSTITUTE

Android: Almacenamiento
Opciones de almacenamiento

- Android proporciona varias opciones para grabar los datos persistentes de las aplicaciones. La solución elegida dependerá de las necesidades específicas, tales como si los datos deberían ser privados a la aplicación o accesibles por otras aplicaciones (y el usuario) o la cantidad de espacio de los datos requieren. Las opciones de almacenamiento de datos son los siguientes:
  - Preferencias compartidas (*Shared Preferences*). Almacenan datos primitivos privados en pares clave-valor.
  - Almacenamiento interno. Almacena datos privados en la memoria del dispositivo.
  - Almacenamiento externo. Almacena datos públicos sobre el almacenamiento externo compartido.
  - Bases de datos SQLite. Almacenamiento estructurado de datos en una base de datos privada.

Android: Almacenamiento
Opciones de almacenamiento

- Android proporciona una manera para publicar, incluso los datos privados a otras aplicaciones - con un proveedor de contenido (*Content Provider*).
- Un proveedor de contenido es un componente opcional que permite el acceso de lectura/escritura a los datos de aplicación, sin perjuicio de las restricciones que se quieran imponer.

http://developer.android.com/guide/topics/providers/content-providers.html

Android: Almacenamiento
Preferencias compartidas

- La clase SharedPreferences proporciona un marco general que permite guardar y recuperar pares clave-valor persistentes de los tipos de datos primitivos.
  - Se puede utilizar para salvar cualquier dato primitivo: booleanos, floats, ints, longs y strings. Estos datos se mantendrá a través de las sesiones de usuario (incluso si muere su aplicación).
- Para obtener un objeto SharedPreferences para la aplicación, se utiliza uno de estos dos métodos:
  - getSharedPreferences() - Utilice esta opción si utiliza varios archivos de preferencias identificadas por su nombre, que se especifica con el primer parámetro. El segundo parámetro indica el modo de operación:
    - MODE_PRIVATE. Sólo nuestra aplicación tiene acceso a estas preferencias.
    - MODE_WORLD_READABLE. Todas las aplicaciones pueden leer estas preferencias, pero sólo la nuestra puede modificarlas.
    - MODE_WORLD_WRITABLE. Todas las aplicaciones pueden leer y modificar estas preferencias.
  - getPreferences() - Utilice esta opción si se utiliza únicamente un archivo de preferencias para la actividad. Sólo se proporcionará el modo de operación.

Android: Almacenamiento
Preferencias compartidas

- Una vez hemos obtenido una referencia a nuestra colección de preferencias, ya podemos obtener, insertar o modificar preferencias utilizando los métodos *get* o *put* correspondientes al tipo de dato de cada preferencia.
- Para leer los valores, se utilizan métodos como getBoolean() o getString()
- Para escribir valores se debe llamar al método edit() para obtener un SharedPreferences.Editor y añadir valores con métodos tales como putBoolean() y putString(). Por último se deben confirmar los nuevos valores con la llamada al método commit()
- las preferencias no se almacenan en ficheros binarios como las bases de datos SQLite, sino en ficheros XML. Estos ficheros XML se almacenan en una ruta que sigue el siguiente patrón:

  /data/data/<paquete java>/shared_prefs/<nombre colección>.xml

Android: Almacenamiento
Almacenamiento interno

- En Android también podremos manipular ficheros tradicionales de una forma muy similar a como se realiza en Java.
- Se pueden guardar los archivos directamente en la memoria interna del dispositivo. De forma predeterminada, los archivos guardados en la memoria interna son privados para la aplicación y otras aplicaciones no pueden acceder a ellos (ni siquiera puede acceder el usuario).
  - Cuando el usuario desinstala la aplicación, se eliminan estos archivos.
- Cuando almacenamos ficheros en la memoria interna debemos tener en cuenta las limitaciones de espacio que tienen muchos dispositivos, por lo que no deberíamos abusar de este espacio utilizando ficheros de gran tamaño.

Android: Almacenamiento
Almacenamiento interno

- Para crear y escribir un archivo privado del almacenamiento interno:
  - Llamar al método openFileOutput() con el nombre del archivo y el modo de acceso, que puede variar entre MODE_PRIVATE para acceso privado desde nuestra aplicación (crea el fichero o lo sobrescribe si ya existe), MODE_APPEND para añadir datos a un fichero ya existente, MODE_WORLD_READABLE para permitir a otras aplicaciones leer el fichero, o MODE_WORLD_WRITABLE para permitir a otras aplicaciones escribir sobre el fichero.
  - El método openFileOutput() devuelve un FileOutputStream a partir del cual ya podremos utilizar los métodos de manipulación de ficheros tradicionales del lenguaje Java.
- Para abrir y leer un archivo privado del almacenamiento interno:
  - Llamar al método openFileInput() con el nombre del archivo. Este método devuelve un FileInputStream a partir del cual ya podremos utilizar los métodos de manipulación de ficheros tradicionales del lenguaje Java.
- La ruta de almacenamiento de los ficheros sigue el siguiente patrón:

  /data/data/<paquete java>/files/<nombre fichero>

Android: Almacenamiento
Almacenamiento interno

- Existe una forma alternativa de almacenar ficheros en la memoria interna del dispositivo que es incluirlos como *recurso* en la propia aplicación. Aunque este método es útil en muchos casos, sólo debemos utilizarlo cuando no necesitemos realizar modificaciones sobre los ficheros, ya que tendremos limitado el acceso a sólo lectura.
- Para incluir un fichero como recurso de la aplicación debemos colocarlo en la carpeta "/res/raw" de nuestro proyecto de Eclipse.
  - Esta carpeta no suele estar creada por defecto, por lo que deberemos crearla manualmente en Eclipse.
- Para acceder al fichero, accederemos en primer lugar a los recursos de la aplicación con el método getResources() y sobre éstos utilizaremos el método openRawResource(*R.raw.nombre_del_fichero*) para abrir el fichero en modo lectura. Este método devuelve un objeto InputStream, que ya podremos manipular como queramos mediante los métodos de Java.

Samsung TECH INSTITUTE

Android: Almacenamiento
Almacenamiento externo

- Todos los dispositivos compatibles con Android soportan "memoria externa" compartida que se puede utilizar para guardar archivos.
  - Puede ser un soporte de almacenamiento extraíble (como una tarjeta SD) o una memoria interna (no extraíble).
- Los archivos guardados en el almacenamiento externo son de lectura global y pueden ser modificados por el usuario cuando permiten el almacenamiento masivo USB para transferir archivos de un ordenador.
  - NOTA: El almacenamiento externo puede no estar disponible y no hay seguridad de que los archivos que se guarden en la memoria externa estén disponibles.
- Para poder leer o escribir archivos en el almacenamiento externo, la aplicación debe adquirir los permisos READ_EXTERNAL_STORAGE o WRITE_EXTERNAL_STORAGE.
  - Si se necesita leer y escribir archivos, sólo hay que fijar el permiso WRITE_EXTERNAL_STORAGE, porque implícitamente requiere acceso de lectura también.

Samsung TECH INSTITUTE

Android: Almacenamiento
Almacenamiento externo

- Antes de realizar cualquier trabajo con el almacenamiento externo, siempre debe llamar getExternalStorageState() para comprobar si el medio está disponible:

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Android: Almacenamiento
Almacenamiento externo

- En general, los archivos que el usuario puede gestionar a través de su aplicación deben ser guardados en un lugar "público" en el dispositivo donde otras aplicaciones pueden tener acceso a ellos y el usuario pueda copiar fácilmente desde el dispositivo.
  - Se debe utilizar cualquiera de los directorios públicos compartidos, como la Music, Pictures o Ringtones.
- Para manejar un archivo (File) en un directorio público se utiliza el método getExternalStoragePublicDirectory(), pasándole el tipo de directorio que desee, como DIRECTORY_MUSIC, DIRECTORY_PICTURES, DIRECTORY_RINGTONES, u otros.
  - Al guardar los archivos en el directorio del tipo de soporte correspondiente, el gestor de medios del sistema puede categorizar adecuadamente los archivos en el sistema.

Android: Almacenamiento
Almacenamiento externo

- Si se quiere manejar archivos que no estén disponibles para otras aplicaciones se debe utilizar un directorio de almacenamiento privado en el almacenamiento externo llamando getExternalFilesDir().
- Este método también toma un argumento para especificar el tipo de subdirectorio (como DIRECTORY_MOVIES). Si no se necesita un directorio específico de comunicación, se pasa *null* para recibir el directorio raíz del directorio privado de su aplicación.
- A partir de Android 4.4, leer o escribir archivos en los directorios privados de la aplicación no requiere los permisos READ_EXTERNAL_STORAGE o WRITE_EXTERNAL_STORAGE.
  - Se puede declarar que el permiso debe solicitarse sólo en las versiones anteriores de Android añadiendo el atributo maxSdkVersion con valor 18.
- Nota: Cuando el usuario desinstala la aplicación, se eliminan este directorio y todo su contenido. Además, el gestor de medios no lee archivos en estos directorios.
  - No son accesibles desde el proveedor de contenido MediaStore.

Android: Almacenamiento
Almacenamiento externo

- Para abrir un archivo (File) que representa el directorio de almacenamiento externo en el que debe guardar sus archivos de caché, llame getExternalCacheDir().
  - Si el usuario desinstala la aplicación, se eliminarán automáticamente estos archivos.
  - Estos archivos son internos a la aplicación, y típicamente no visibles para el usuario como medios.
- También se puede acceder a un directorio de caché en un almacenamiento externo secundario (si está disponible) llamando ContextCompat.getExternalCacheDirs().
- Nota: Para preservar el espacio de archivos y mantener el rendimiento de la aplicación, es importante que manejar cuidadosamente los archivos de caché y eliminar los que ya no son necesarios en todo el ciclo de vida de la aplicación.

Android: Almacenamiento
Utilizando SQLite

- Android proporciona soporte completo para bases de datos SQLite.
- Las bases de datos creadas dentro de una aplicación podrán ser accedidas por cualquier clase de la misma, pero no por clases externas a la aplicación.
- El método recomendado para crear una nueva base de datos SQLite es crear una subclase de SQLiteOpenHelper y sobreescribir el método onCreate() para ejecutar los comandos SQLite necesarios para crear tablas en la base de datos.
- A continuación, se obtiene una instancia de la implementación de SQLiteOpenHelper y se llama a los métodos getWritableDatabase () o getReadableDatabase () para escribir o leer de la base de datos respectivamente.
  - Estos métodos devuelven un objeto SQLiteDatabase que representa la base de datos y proporciona métodos SQLite para operar con ella.

Android: Almacenamiento
Utilizando SQLite

- subclase de SQLiteOpenHelpe:

```
public class DictionaryOpenHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 2;
    private static final String DICTIONARY_TABLE_NAME = "dictionary";
    private static final String DICTIONARY_TABLE_CREATE =
                "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +
                KEY_WORD + " TEXT, " +
                KEY_DEFINITION + " TEXT);";

    DictionaryOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DICTIONARY_TABLE_CREATE);
    }
}
```

Android: Almacenamiento
Utilizando SQLite

- Se pueden ejecutar consultas SQLite utilizando el método query() de SQLiteDatabase, que acepta varios parámetros de consulta, tales como la tabla de consulta, la proyección, la selección, las columnas, la agrupación, y otros.
- Para consultas complejas, como las que requieren los alias de columna, se debe utilizar SQLiteQueryBuilder, que proporciona varios métodos prácticos para la creación de consultas.
- Cada consulta SQLite devuelve un Cursor que apunta a todos los registros encontrados por la consulta. El cursor es siempre el mecanismo con el que se puede navegar por los resultados de una consulta de base de datos y leer filas y columnas.
- El SDK de Android incluye una herramienta de base de datos sqlite3 que permite mostrar los contenidos de las tablas, ejecutar comandos SQL, y realizar otras funciones en bases de datos SQLite.

Android: Almacenamiento
Proveedores de contenido

- Los proveedores de contenido administran el acceso a un conjunto estructurado de datos. Encapsulan los datos, y proporcionan mecanismos para la definición de seguridad de los mismos. Los proveedores de contenido son la interfaz estándar que conecta los datos en un proceso con el código que se ejecuta en otro proceso.
- Para acceder a los datos en un proveedor de contenido, se utiliza el objeto ContentResolver en el contexto de la aplicación para comunicarse tanto con el proveedor como con el cliente.
- El objeto ContentResolver se comunica con el objeto de proveedor, una instancia de la clase que implementa ContentProvider. El objeto de proveedor recibe solicitudes de datos de clientes, realiza la acción solicitada, y devuelve los resultados.
- No es necesario para desarrollar un proveedor si no se tiene intención de compartir los datos con otras aplicaciones. Sin embargo, será necesario para proporcionar sugerencias de búsqueda personalizada en la propia aplicación o si desea copiar y pegar datos o archivos complejos de la aplicación a otras aplicaciones.
- Android incluye proveedores de contenido que manejan datos, tales como audio, vídeo, imágenes y datos de contacto personal. Con algunas restricciones, estos proveedores son accesibles para cualquier aplicación de Android.

Android: Almacenamiento
Proveedores de contenido

- Un proveedor de contenidos presenta los datos a aplicaciones externas como una o más tablas que son similares a las tablas que se encuentran en una base de datos relacional. Una fila representa una instancia de un tipo de datos que el proveedor recopila, y cada columna de la fila representa una pieza individual de los datos recopilados para una instancia.
- Una aplicación tiene acceso a los datos de un proveedor de contenido con un objeto de cliente ContentResolver. Este objeto tiene métodos que llaman a métodos con nombres idénticos en el objeto de proveedor, una instancia de una de las subclases concretas de ContentProvider. Los métodos del ContentResolver proporcionan las funciones "CRUD" básicas (crear, recuperar, actualizar y eliminar) de almacenamiento persistente.
- El objeto ContentResolver en el proceso de la aplicación cliente y el objeto ContentProvider en la aplicación que posee el proveedor manejan de forma automática la comunicación entre procesos. ContentProvider también actúa como una capa de abstracción entre el repositorio de datos y la apariencia externa de los datos en forma de tablas.

Android: Almacenamiento
Proveedores de contenido

- Una URI de contenido es una URI que identifica los datos de un proveedor. Incluyen el nombre simbólico del proveedor (su autoridad) y un nombre que apunta a una tabla (la ruta). Cuando se llama a un método de cliente para acceder a una tabla en un proveedor, la URI de la tabla es uno de los argumentos.
- El objeto ContentResolver analiza la autoridad de la URI, y la utiliza para "resolver" el proveedor mediante la comparación con una tabla del sistema de proveedores conocidos. El ContentResolver puede entonces enviar los argumentos de la consulta al proveedor correcto.
- El ContentProvider utiliza la parte de la ruta de la URI de contenido para elegir la tabla a acceder. Un proveedor por lo general tiene un camino para cada tabla que publica.

Android: Almacenamiento
Proveedores de contenido

- Consultando datos, pasos básicos:
  1. Solicitar el permiso de acceso de lectura para el proveedor.
     - Para recuperar los datos de un proveedor, la aplicación debe "leer permiso de acceso" para el proveedor. No se puede solicitar este permiso en tiempo de ejecución hay que especificarlo en el manifiesto, mediante el elemento <uses-permission> y el nombre exacto permiso definido por el proveedor.
     - Para encontrar el nombre exacto del permiso de acceso de lectura para el proveedor que está utilizando, así como los nombres de otros permisos de acceso utilizados por el proveedor, hay que buscar en la documentación del proveedor.

Android: Almacenamiento
Proveedores de contenido

- Permisos.
  - Una aplicación de un proveedor puede especificar los permisos que otras aplicaciones deben tener para poder acceder a los datos del proveedor. Estos permisos aseguran que el usuario sepa qué datos que aplicaciones tendrán acceso.
  - Sobre la base de los requisitos del proveedor, otras aplicaciones solicitan los permisos que necesitan con el fin de obtener acceso al proveedor. Los usuarios finales ven los permisos solicitados al instalar la aplicación.
  - Si la aplicación de un proveedor no especifica ningún permiso, el resto de aplicaciones no tienen acceso a los datos del proveedor. Sin embargo, los componentes de la aplicación del proveedor siempre tienen acceso total de lectura y escritura, con independencia de los permisos especificados.
  - Para obtener los permisos necesarios para acceder a un proveedor, una aplicación lo solicita con un elemento <uses-permission> en su archivo de manifiesto. Cuando el Administrador de paquetes Android instala la aplicación, el usuario debe aprobar todos los permisos de la aplicación solicita sino se cancela el proceso de instalación.

Android: Almacenamiento
Proveedores de contenido

2. Definir el código que envía una consulta al proveedor.

query() argument	SELECT keyword/parameter	Notes
`Uri`	`FROM table_name`	`Uri` maps to the table in the provider named *table_name*.
`projection`	`col,col,col,...`	`projection` is an array of columns that should be included for each row retrieved.
`selection`	`WHERE col = value`	`selection` specifies the criteria for selecting rows.
`selectionArgs`	(No exact equivalent. Selection arguments replace `?` placeholders in the selection clause.)	
`sortOrder`	`ORDER BY col,col,...`	`sortOrder` specifies the order in which rows appear in the returned `Cursor`.

Samsung TECH INSTITUTE

Android: Almacenamiento
Proveedores de contenido

3. Mostrar los resultados de la consulta.
   - El método de cliente ContentResolver.query() siempre devuelve un Cursor que contiene las columnas especificadas por la proyección de la consulta para las filas que coinciden con los criterios de selección de la consulta.
   - Si no hay filas coinciden con los criterios de selección, el proveedor devuelve un objeto Cursor para el que Cursor.getCount() es 0 (un cursor vacío).
   - Si se produce un error interno, los resultados de la consulta dependen del proveedor en particular. Se puede optar por devolver null, o se puede producir una excepción.
   - Como un Cursor es una "lista" de filas, una buena forma de mostrar el contenido de un cursor es vincular a un ListView a través de un SimpleCursorAdapter.
     - Nota: Para enlazar un ListView con un Cursor, el Cursor debe contener una columna denominada _ID.

Samsung TECH INSTITUTE

Android: Almacenamiento
Proveedores de contenido

- Insertando datos.
  - Para insertar datos en un proveedor, se llama al método ContentResolver.insert(). Este método inserta una nueva fila en el proveedor y devuelve un URI de contenido de esa fila.
  - Los datos para la nueva fila van en un solo objeto ContentValues, que es similar en su forma a un Cursor de una sola fila. Las columnas de este objeto no necesitan tener el mismo tipo de datos, y si no se desea especificar un valor, se puede establecer una columna a null utilizando ContentValues.putNull().
- Actualizando datos.
  - Para actualizar una fila, se utiliza un objeto ContentValues con los valores actualizados tal como se hace con una inserción, y con los criterios de selección tal como se hace con una consulta. El método de cliente que se utiliza es ContentResolver.update(). Si se desea borrar el contenido de una columna, se establece el valor en null.

Android: Almacenamiento
Proveedores de contenido

- Borrando datos.
  - La eliminación de filas es similar a la recuperación de datos de filas: se especifican criterios de selección de las filas que se desea eliminar y el método de cliente ContentResolver.delete() devuelve el número de filas eliminadas.

Android: Almacenamiento
Proveedores de contenido

- Forma alternativas de proporcionar acceso.
  - **Acceso por lotes**: puede crear una serie de llamadas de acceso con métodos de la clase ContentProviderOperation y, a continuación, aplicar con ContentResolver.applyBatch().
  - **Consultas asíncronas**: Se deben hacer consultas en un subproceso independiente. Una forma de hacer esto es utilizar un objeto CursorLoader.
  - **Acceso a los datos a través de las intenciones**: Se puede acceder a los datos en un proveedor de contenidos, incluso si no se tienen los permisos de acceso, mediante el envío de un intento de una aplicación que tiene los permisos y recibir de vuelta un intento de resultados que contiene permisos "URI". Son permisos temporales mientras que la actividad que los ha otorgado no finalice.

Samsung TECH INSTITUTE