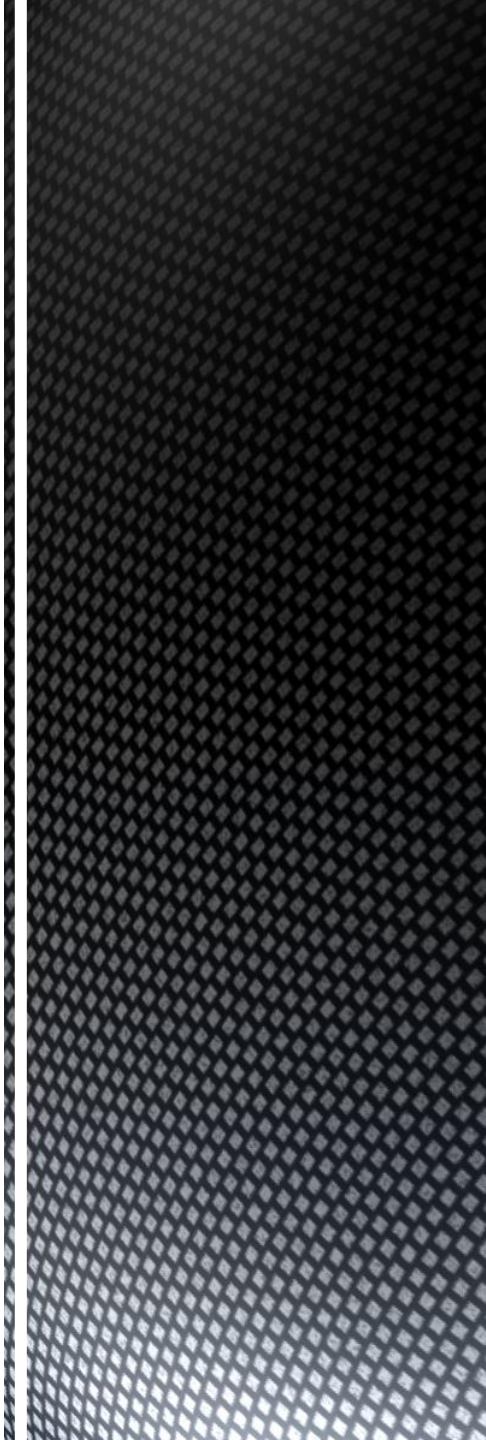
Android

Ficheros – Internal Storage



Ficheros

Android permite almacenar ficheros en la memoria interna, en un directorio reservado para la aplicación y sólo accesible por esta.

/data/data/<app_package>/files

La memoria interna siempre está disponible pero es limitada.

Al desinstalar la aplicación se borran los ficheros de ésta.

Sólo el usuario linux que se crea para la aplicación tiene acceso a los ficheros.

Saving Files

Primero hay que saber cual es el directorio de almacenamiento interno.

Esto se consigue con getFilesDir() que devuelve un objeto de tipo File que es una representación abstracta del sistema de ficheros mediante una ruta.

Para crear un fichero en la ruta interna podemos hacer uso de:

- `File file = new File(context.getFilesDir(), filename);`
- O bien con los métodos, también de la clase Context

  openFileInput() y openFileOutput() que devuelven un FileInputStream o un FileOutputStream, es decir un flujo de lectura o escritura en el directorio interno, un flujo de lectura/escritura de/hacia un fichero.

Debes cerrar siempre los ficheros con close()

Ficheros

Utilizar una construcción try-catch para determinar si tenemos acceso o no al medio y gestionar los errores

```
String fichero = "fichero.txt";
String texto = "texto almacenado";
FileOutputStream fos;

try {
  fos = openFileOutput(fichero,Context.MODE_PRIVATE);
  fos.write(texto.getBytes());
  fos.close();
  }
catch (FileNotFoundException e) {
  Log.e("Mi Aplicación",e.getMessage(),e);
  }
catch (IOException e) {
  Log.e("Mi Aplicación",e.getMessage(),e);
  }
```

MODE_PRIVATE
: Sólo accesible por la aplicación que lo creó

MODE_APPEND
: Para añadir a uno ya existente

En el stream se almacenan Bytes.

Se puede utilizar también las clases:
OutputStreamWriter
InputStreamReader

que permiten leer el fichero por bloques, y convierte a bytes automáticamente al grabar y a caracteres al leer.

Otros métodos útiles:

- getFilesDir() Devuelve la ruta absoluta donde se guardan los ficheros
- getDir() Crea un directorio en tu almacenamiento interno o lo abre
- deleteFile() Borra un fichero
- fileList() devuelve un array con los ficheros almacenados por tu app.

Ficheros

Usando Output / Input Streams

InputStreamReader

- Convierte un stream de bytes en caracteres
- Necesita un InputStream para leer de él
- Se puede leer de él carácter a carácter con read()
- O también un conjunto de caracteres con read(char[] buffer, int offset, int count)

BufferedReader

- Encapsula la lectura de un Reader (un InputStramReader) proporcionando un buffer de lectura de 8Kb lo que acelera operaciones de lectura.
- Nos permite, a parte de los read, leer línea a línea con readLine()

Ficheros

Usando Output / Input Streams

```
private void writeToFile(String data,Context context) {
    try {
        OutputStreamWriter outputStreamWriter =
                    new OutputStreamWriter(context.openFileOutput("config.txt", Context.MODE_PRIVATE));
        outputStreamWriter.write(data);
        outputStreamWriter.close();
    }
    catch (IOException e) {
        Log.e("Exception", "File write failed: " + e.toString());
    }
}
```

```
private String readFromFile(Context context) {
    String ret = "";
    try {
        InputStream inputStream = context.openFileInput("config.txt");
        if ( inputStream != null ) {
            InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
            BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
            String receiveString = "";
            StringBuilder stringBuilder = new StringBuilder();

            while ( (receiveString = bufferedReader.readLine()) != null ) {
                stringBuilder.append(receiveString);
            }

            inputStream.close();
            ret = stringBuilder.toString();
        }
    }
    catch (FileNotFoundException e) {
        Log.e("login activity", "File not found: " + e.toString());
    } catch (IOException e) {
        Log.e("login activity", "Can not read file: " + e.toString());
    }
    return ret;
}
```

Ficheros

El siguiente código muestra cómo utilizando un BufferedReader podemos leer un fichero línea por línea haciendo uso de su método readLine().

```
try {  // open the file for reading
  InputStream instream = new FileInputStream("myfilename.txt");

  // if file the available for reading
  if (instream != null) {
     // prepare the file for reading
     InputStreamReader inputreader = new InputStreamReader(instream);
     BufferedReader buffreader = new BufferedReader(inputreader);

     String line;

     // read every line of the file into the line-variable, on line at the time
     do {
        line = buffreader.readLine();
        // do something with the line
        } while (line != null);

     }
  }
catch (Exception ex) {
    // print stack trace.
  }
finally {
  // close the file.
  instream.close();
  }
```

Ejercicio E0601_SavingText