PDM Avanzado Datos Persistentes Arquivos

Sumario

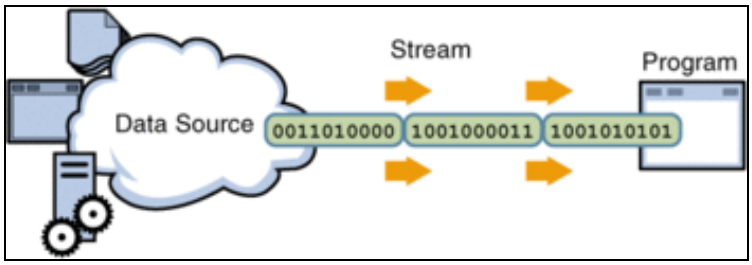
Introdución

- O tratamento dos ficheiros en Android é idéntico a Java.
- Recomendamos que se revise o apartado do curso de iniciación adicado ao tratamento de Ficheiros

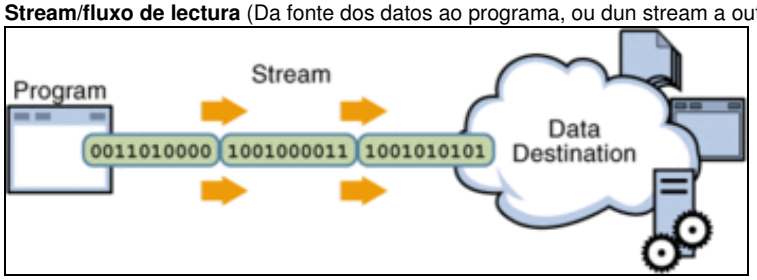
Introdución á E/S en Java

- O paquete **java.io** contén as clases para manipular a E/S.
- En java a entrada/saída xestionase a través de **streams (fluxos)** , e estes poden interactuar cun teclado, a consola, un porto, un ficheiro, outro stream, etc.
- Todo stream ten un orixe e un destino.

- Streams
-



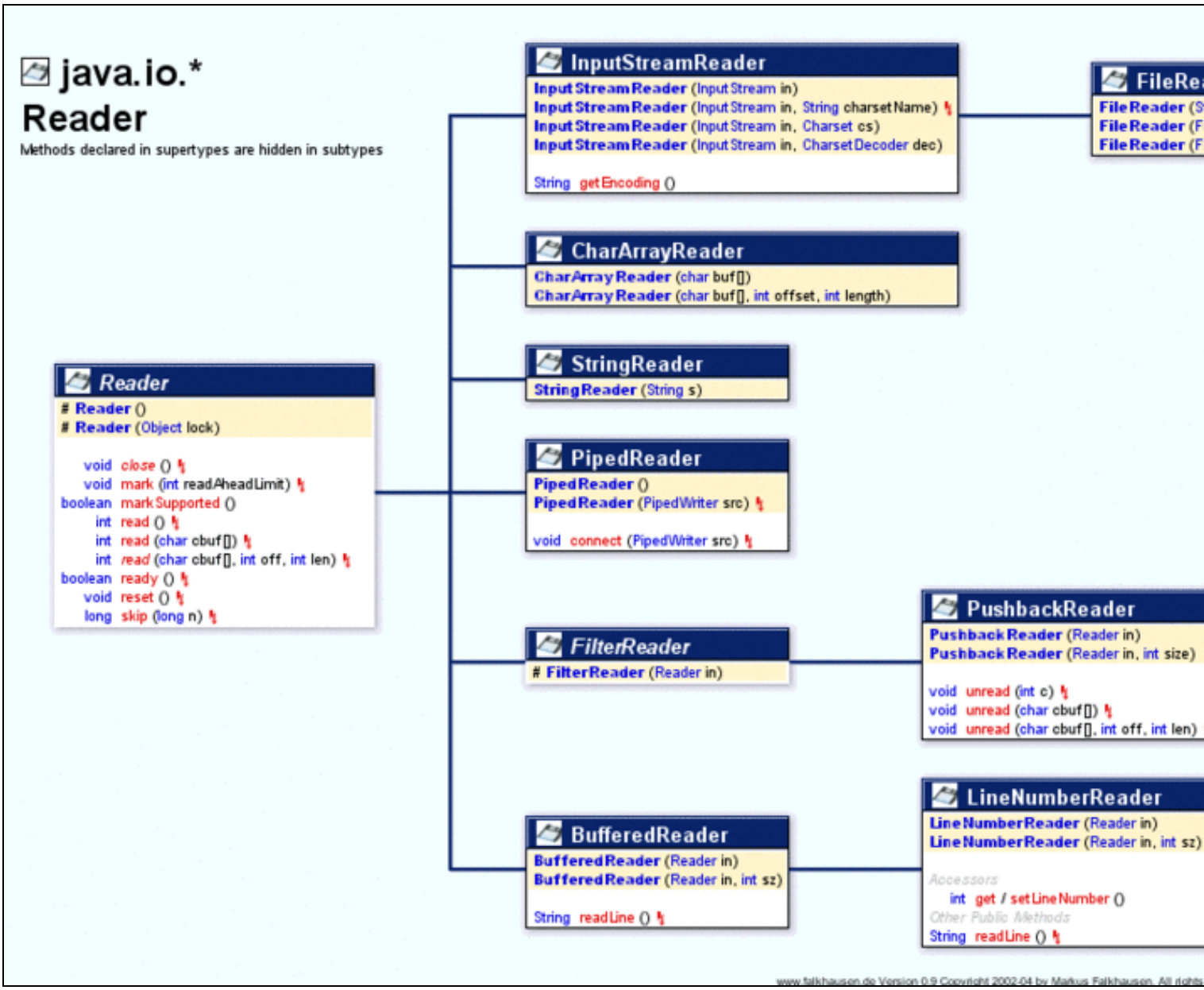
Stream/fluxo de lectura (Da fonte dos datos ao programa, ou dun stream a outro stream).

- 

Stream/fluxo de escritura (Do programa ao destino dos datos, ou dun stream a outro stream).

- O fluxo máis básico de E/S son os fluxos de bytes, pero un fluxo de bytes pode ser a entrada doutro fluxo máis complexo, até chegar a ter fluxos de caracteres, e de buffers e o mesmo á inversa.

- Para aquel usuario que desexe repasar ou afondar na E/S en java déixanse os seguintes enlaces:
    - Curso de Java nos Manuais do IES San Clemente: Entrada/Saída
    - Diagramas moi gráficos (valga a redundancia) das xerarquías de clases de E/S, onde se poden ver as clases, atributos, construtores, métodos, etc dun modo moi claro:
    - http://www.falkhausen.de/en/diagram/html/java.io.Writer.html (Neste caso da xerarquía writer).
    - A modo de exemplo amósase un exemplo de diagrama da xerarquía *Reader*:
    - Observar no diagrama como os construtores da clase **InputStreamReader** reciben como parámetro outro stream/fluxo de tipo **InputStream** (Que está noutra xerarquía).


java.io.*
Reader
Methods declared in supertypes are hidden in subtypes
Reader
Reader ()
Reader (Object lock)
void close () ↯
void mark (int readAheadLimit) ↯
boolean markSupported ()
int read () ↯
int read (char cbuf[]) ↯
int read (char cbuf[], int off, int len) ↯
boolean ready () ↯
void reset () ↯
long skip (long n) ↯
InputStreamReader
InputStreamReader (InputStream in)
InputStreamReader (InputStream in, String charsetName) ↯
InputStreamReader (InputStream in, Charset cs)
InputStreamReader (InputStream in, CharsetDecoder dec)
String getEncoding ()
FileRe
FileReader (S
FileReader (F
FileReader (F
CharArrayReader
CharArrayReader (char buf[])
CharArrayReader (char buf[], int offset, int length)
StringReader
StringReader (String s)
PipedReader
PipedReader ()
PipedReader (PipedWriter src) ↯
void connect (PipedWriter src) ↯
FilterReader
FilterReader (Reader in)
PushbackReader
PushbackReader (Reader in)
PushbackReader (Reader in, int size)
void unread (int c) ↯
void unread (char cbuf[]) ↯
void unread (char cbuf[], int off, int len)
LineNumberReader
LineNumberReader (Reader in)
LineNumberReader (Reader in, int sz)
Accessors
int get / setLineNumber ()
Other Public Methods
String readLine () ↯
BufferedReader
BufferedReader (Reader in)
BufferedReader (Reader in, int sz)
String readLine () ↯
www.falkhausen.de Version 0.9 Copyright 2002-04 by Markus Falkhausen. All rights

Introdución aos ficheiros en Android

- Os ficheiros en Android poden servirnos para almacenar información de modo temporal, para pasar información entre dispositivos, para ter unha "mini" base de datos, etc.
- En Android os ficheiros poden almacenarse en tres sitios (e dentro dun deles en 2 directorios distintos).
    - ♦ Na **propia aplicación**:
        - A modo de recurso (como cando incluímos unha imaxe): **/res/raw/ficheiro...** (raw significa cru). Neste caso referenciamos o recurso a través da clase R.
        - No cartafol /assets/. A diferenza do anterior non se xera ningún identificador en R e debemos referenciar os recursos gardados neste cartafol a través da clase AssetManager.
          Neste lugar podemos crear unha estrutura (con cartafoles) cousa que en /res/raw/ non podemos facer.

      Podemos facer operacións de <u>só de lectura</u>.
    - ♦ Na **memoria interna**, no subdirectorio *files* da carpeta da aplicación: **/data/data/paquete_java/files/ficheiro...**.

      Podemos facer operacións de lectura / **escritura**.
    - ♦ Na **tarxeta SD**, se existe, en 2 posibles subdirectorios:
        - ◊ **/storage/sdcard/directorio que indique o programador, se indica/ficheiro...**
        - ◊ **/storage/sdcard/Android/data/paquete_java/files/ficheiro...** (Algo parecido á memoria interna). Deste xeito se se desinstala a aplicación, tamén se borraría o ficheiro automaticamente, cousa que non pasaría no caso anterior

      Podemos facer operacións de lectura / **escritura**.

Primeiro debemos decidir onde gardaremos os datos. Na memoria interna ou na tarxeta externa. Cada opción ten as súas vantaxes e desvantaxes:

- Interna:
    - ◊ Está sempre dispoñible.
    - ◊ Por defecto os datos gardados están só dispoñibles para a aplicación.
    - ◊ Cando o usuario desinstala a aplicación os datos son borrados.
- Externa:
    - ◊ Pode non estar dispoñible (o usuario pode quitar a tarxeta ou non tela).
    - ◊ Poden acceder os datos fora da nosa aplicación.
    - ◊ Os datos gardados aquí permanecen. Unicamente se borran se os gardamos no cartafol indicado polo método getExternalFilesDir().

A clase File e a clase Environment

Veremos máis adiante que para referenciar a arquivos que se atopan na memoria interna ou sd externa imos facer uso da clase File.

Un obxecto da clase File vai poder 'apuntar' a un cartafol / arquivo que se atope na memoria interna / sd externa do dispositivo.

Para apuntar necesitaremos enviarlle como parámetro a ruta a dito arquivo / cartafol desta forma:

```
File arquivo = new File("/sdcard/pictures/imaxe.jpg");
```

No exemplo estariamos apuntado a unha imaxe que se atopa na tarxeta sd externa no cartafol /pictures.

Para indicar as rutas, en vez de escribilas directamente, como fixemos no exemplo, faremos uso da clase Environment.

Dita clase vainos devolver rutas a sitios predeterminados do noso dispositivo Android.

Por exemplo:

- **Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)**: Cartafol recomendado para gardar fotos (dará como resultado /sdcard/pictures/ ou o seu equivalente).
- **Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MOVIES)**: Cartafol recomendado para gardar vídeos.
- **Environment.getExternalStorageDirectory()**: Cartafol do almacenamento externo en Android.

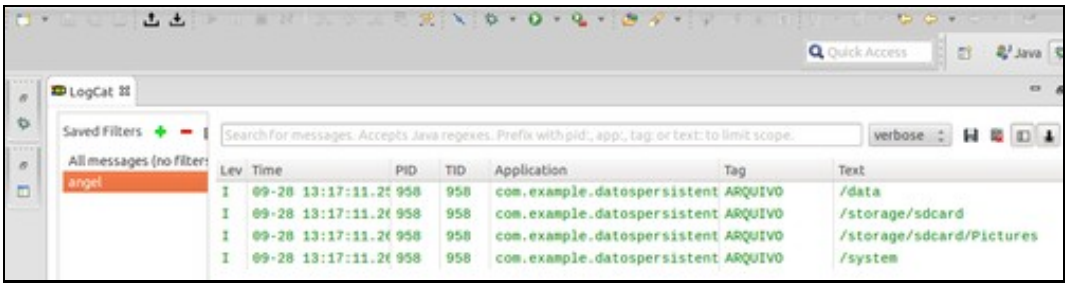
As chamadas anteriores devolven un obxecto da clase **File**.

Se queremos obter a ruta en forma de cadea debemos chamar ao método **getAbsolutePath()** da forma seguinte.

Por exemplo:

```
Log.i("ARQUIVO",Environment.getDataDirectory().getAbsolutePath());
Log.i("ARQUIVO",Environment.getExternalStorageDirectory().getAbsolutePath());
Log.i("ARQUIVO",Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES).getAbsolutePath());
Log.i("ARQUIVO",Environment.getRootDirectory().getAbsolutePath());
```

Dará como resultado:



Existen outros métodos relacionados coa clase Environment como por exemplo:

- **Environmet.getExternalStorageState()** devolve un valor que nos sirve para saber se a tarxeta sd externa está dispoñible en modo lectura/escritura:

```
if (!Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)){
Log.e("ARQUIVO","TARXETA NON MONTADA");
}
```

- **Environment.isExternalStorageEmulated()** indica se a tarxeta sd é emulada:

```
if (Environment.isExternalStorageEmulated()){
        Log.i("ARQUIVO","TARXETA EMULADA");
}
```

A partires dun obxecto da clase File podemos facer as seguintes operacións (dependendo se apuntamos a un cartafol ou a un arquivo):

- **String getAbsolutePath()**: devolve a ruta do cartafol / arquivo en forma de cadea.
- **String getName()**: devolve o nome do arquivo ou cartafol do obxecto file.
- **Boolean IsDirectory()**: indica se o obxecto file 'apunta' a un cartafol.
- **Boolean isFile()**: indica se o obxecto file 'apunta' a un arquivo.
- **Long length()**: devolve o tamaño en byte's do arquivo.
- **String[] list()**: devolve un array de cadeas que representan a cada un dos arquivos que se atopan no cartafol indicado polo obxecto file.
- **File[] listFiles()**: devolve un array de arquivos que representan a cada un dos arquivos que se atopan no cartafol indicado polo obxecto file.
- **Boolean mkdir()**: crear un cartafol indicado polo obxecto File, asumindo que o cartafol pai existe.

  Por exemplo: /storage/sdcard/arquivos/datos

  Se fago o mkdir só vai crear o cartafol 'datos'.

- **Boolean mkdirs()**: crea o cartafol indicado polo obxecto File e tamén todos os seus pais se son necesarios.

  Por exemplo: /storage/sdcard/arquivos/datos

Se fago o mkdirs crearía o cartafol 'arquivos' e o cartafol 'datos'.

- **Boolean exists()**: devolve true se a ruta ou arquivo existe.

Nota: Unha constante que tamén debe usarse é **File.separator** que representa o caracter de separación entre diferentes rutas (por exemplo en Linux é /).

Traballando con cartafoles

Para crear un novo cartafol teremos que engadir á nova ruta a ruta existente: Isto o podemos facer de dúas formas:

- ```
  File nova_ruta = new File(Environment.getExternalStorageDirectory(),"NOME_CARTAFOL_CREAR");
  nova_ruta.mkdirs();
  ```

  Nesta forma non enviamos o separador (/).

- ```
  String novopath=Environment.getExternalStorageDirectory().getAbsolutePath() + File.separator  + "NOME_CARTAFOL_CREAR");
  File nova_ruta = new File(novopath);
  nova_ruta.mkdirs();
  ```

  Fixarse como nesta segunda forma temos que enviarlle o separador de arquivos (File.separator = "/").

Un exemplo para borrar un cartafol:

```
String borrarCartafol=Environment.getExternalStorageDirectory().getAbsolutePath() + File.separator  + "NOME_CARTAFOL_BORRAR");
File borrar = new File(borrarCartafol);
borrar.delete();
```

Traballando con arquivos

O funcionamento é igual ao anterior, pero indicando o nome do arquivo a borrar / ler.

Por exemplo:

```
String novopath=Environment.getExternalStorageDirectory().getAbsolutePath() + File.separator  + "ARQUIVOS");
File arquivo = new File(novopath,"arquivo_a_leer.txt");
```

Neste exemplo estamos a 'apuntar' a un arquivo de nome 'arquivo_a_leer.txt' se atopa na tarxeta SD Externa no cartafol ARQUIVOS.

Fixarse que isto non significa que dito arquivo exista. Podemos ter o obxecto da clase File para crear dito arquivo (o veremos despois).

Por exemplo, podemos comprobar si o arquivo existe:

```
String novopath=Environment.getExternalStorageDirectory().getAbsolutePath() + File.separator  + "ARQUIVOS");
File arquivo = new File(novopath,"arquivo_a_leer.txt");
if (arquivo.exists()){
   // PODEMOS LER
}
```

Unha vez temos a referencia ao obxecto **File** xa podemos ler / escribir.

Accedendo aos ficheiros

Neste punto imos aprender como ler datos dende os diferentes lugares de almacenamento descritos no punto anterior.

Existen outras moitas formas de facelo, nos imos explicar unha delas.

Dependendo se queremos ler textos ou bytes (por exemplo, no caso dunha imaxe), necesitamos obxectos pertencentes a clases diferentes:

- Texto: O que necesitamos é obter un obxecto da clase InputStreamReader no caso da lectura.

  Este obxecto vainos permitir percorrer todo o arquivo lendo datos nel.

- Bytes (por exemplo unha imaxe, unha base de datos): O que necesitamos é obter un obxecto da clase InputStream no caso da lectura.

Como obtemos estes obxectos ?

Dependendo do lugar onde estea almacenada a información.

Accedendo a recursos internos (cartafol raw)

Neste caso o recurso é compilado xunto co resto de recursos e podemos acceder a eles coa clase R.raw.

Nota: Lembrar que en /res/raw/, ao igual que todo o que se atopa en /res/, o nome dos recursos só poden levar letras minúsculas e números así como guión baixo.

Binarios

Caso de imaxes ou bases de datos, por exemplo.

Neste caso podemos obter a referencia ao obxecto da clase InputStream da forma:

```
InputStream is_raw = getResources().openRawResource(R.raw.imaxe);
```

Sendo R.raw.imaxe unha imaxe gardada en /res/raw/ de nome "imaxe.jpg".

Texto

Caso de arquivos de texto como arquivos xml, txt, ou calquera outro que poida editarse cun editor de textos.

Neste caso podemos obter a referencia ao obxecto da clase InputStreamReader da forma:

```
InputStream is_raw = getResources().openRawResource(R.raw.arquivotexto);
InputStreamReader isreader_raw = new InputStreamReader(is_raw);
```

Sendo R.raw.arquivotexto un arquivo de texto gardado en /res/raw/ de nome "arquivotexto.txt".

Accedendo ao cartafol asset

Neste cartafol se gardan normalmente recursos que non necesitamos 'compilar' coa aplicación, coma poden ser gráficos, bases de datos,...

Permite crear unha estrutura de cartafoles como se fose un sistema de ficheiros (cousa que no cartafol raw non podemos facer).

Binarios

```
InputStream is_assets = getAssets().open("imagen.jpg");
```

Texto

```
InputStream is_assets = getAssets().open("texto.txt");
InputStreamReader isreader_assets = new InputStreamReader(is_assets);
```

Accedendo á memoria interna

Se queremos gardar algo na memoria interna do dispositivo teriamos que facer uso do seguinte método:

- **getFilesDir()**: que devolve un obxecto da clase File, o cal 'apunta' a 'data/data/package_name/files'.

Lembrar que nesta memoria si podemos escribir datos.

Binarios

- Lectura

```
File ruta = getFilesDir();
File arquivo = new File(ruta,"arquivo.txt");
try {
InputStream os_interna = new FileInputStream(arquivo);
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

- Escritura

```
File ruta = getFilesDir();
File arquivo = new File(ruta,"arquivo.txt");
try {
OutputStream os_interna_out = new FileOutputStream(arquivo);
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

Texto

- Lectura

Pode ser así:

```
File ruta = getFilesDir();
File arquivo = new File(ruta,"arquivo.txt");
try {
     FileInputStream fis_sd = new FileInputStream(arquivo);
                  InputStreamReader isreader_sd = new InputStreamReader(fis_sd);
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

Ou así:

```
FileInputStream fis_sd=null;
try {
FileInputStream fis_sd = openFileInput(getFilesDir().getAbsolutePath() + File.separator + "arquivo.txt");
                        InputStreamReader isreader_sd = new InputStreamReader(fis_sd);
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

- Escritura

É igual ao anterior cambiando FileInputStream / InputStreamReader por FileOutputStream / OutputStreamWriter

Pode ser así:

```
File ruta = getFilesDir();
```

```
File arquivo = new File(ruta,"arquivo.txt");
try {
     FileOutputStream fos_sd = new FileOutputStream(arquivo);
                    OutputStreamWriter iswriter_sd = new OutputStreamWriter(fos_sd);
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

Ou así:

```
try {
    FileOutputStream fos_sd = openFileOutput(getFilesDir().getAbsolutePath() + File.separator + "arquivo.txt",MODE_PRIVATE);
            OutputStreamWriter iswriter_sd = new OutputStreamWriter(fos_sd);
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

Engadir / Crear arquivo novo

No caso de escribir nun arquivo podemos abrilo para crealo ou para engadir novo contido.

Nos exemplos anteriores sempre sobreescribimos o arquivo.

Necesitaremos crear un obxecto da clase:

◊ **FileOutputStream(ficheiro)** ou **FileOutputStream(ficheiro,engadir)**, onde:
- **Ficheiro:** é a ruta ao ficheiro que lle temos que indicar nós ou ben a *lume* (/data/data/.../files) ou facendo uso do método: **getFilesDir()**, que devolve a ruta ao directorio **files** da aplicación.
- **Engadir**: booleano que indica se o ficheiro se abre en modo sobrescritura ou append.

Outra forma de obter un FileOutputStream é facendo uso do método **openFileOutput(ficheiro, modo_acceso_ao_ficheiro).**

Este método abre o ficheiro indicado no modo indicado, que pode ser:

◊ **MODE_PRIVATE**: para acceso privado dende a nosa app, e non dende outras.
◊ **MODE_APPEND**: para engadir datos a un ficheiro existente
◊ **MODE_WORLD_READABLE**: permitir que outras app lean o ficheiro
◊ **MODE_WORLD_WRITEABLE**: permitir que outras app lean/escriban o ficheiro

◊ O método devolve un stream asociado ao ficheiro de tipo **FileOutputStream**, a partir de aquí xa podemos operar con ese fluxo como o faríamos en Java.
◊ O método crea o ficheiro no cartafol: /**data**/**data**/**paquete_java**/**files**/**ficheiro**.

Accedendo á tarxeta SD Externa

Memoria Externa: permisos de escritura na tarxeta SD

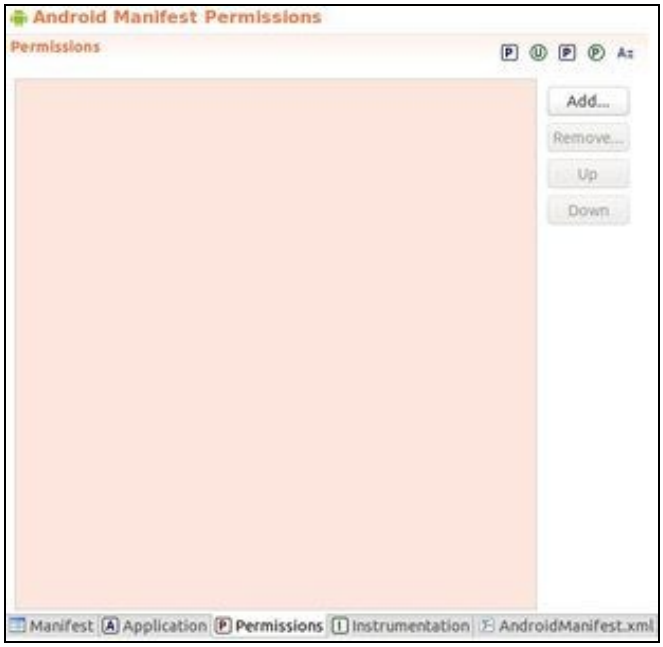
- Se imos ler na tarxeta SD:

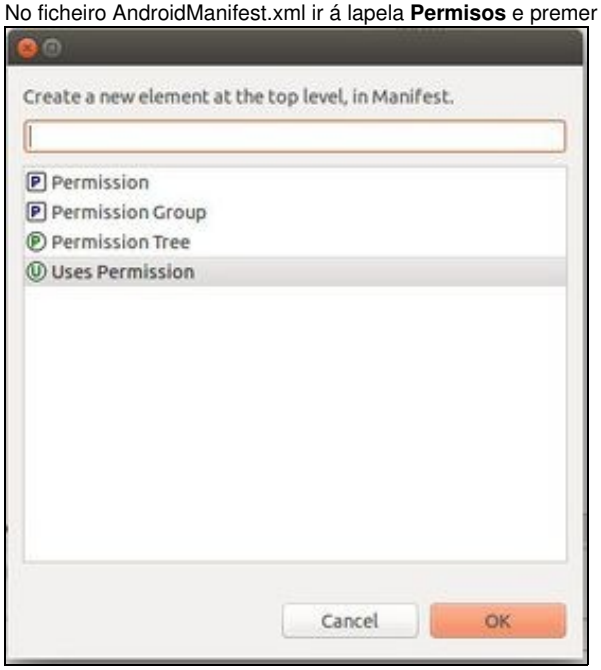
◊ Se a versión do S.O. Android é inferior á 4.1 non precisamos ningún permiso.
◊ Se a versión do S.O. Android é superior ou igual á 4.1 debemos engadir o permiso: **<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>**.

- Se imos escribir na tarxeta SD:

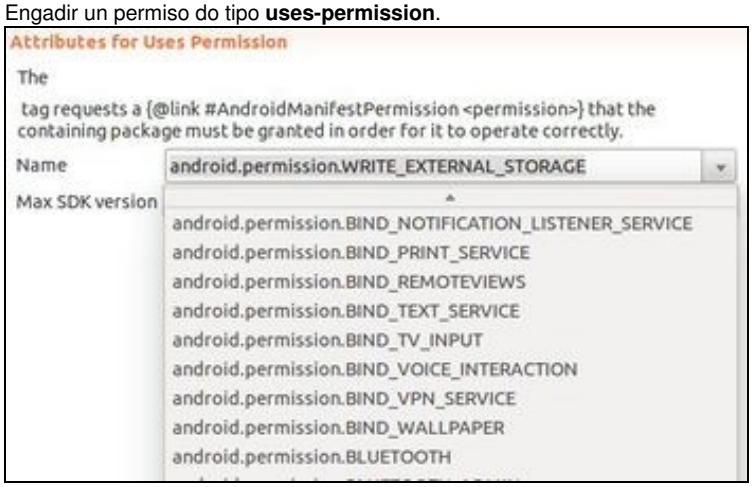
◊ Se a versión do S.O. Android é inferior á 4.4 o permiso é: **<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>** .
◊ Se a versión do S.O. Android é a 4.4 ou superior. Podemos poñer o mesmo permiso anterior pero as aplicacións dispoñen dun cartafol para escribir na SD (cartafol Android/data/paquete/) sen necesidade de ter o permiso anterior.

Os permisos necesarios son postos no ficheiro **AndroidManifest.xml** da aplicación.

- Permiso escritura na Memoria Externa
- 

No ficheiro AndroidManifest.xml ir á lapela **Permisos** e premer en Engadir.
- 

Engadir un permiso do tipo **uses-permission**.

- 

Engadir o permiso: **android.permission.WRITE_EXTERNAL_STORAGE**.

- 

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.u4_12_ficheirosd"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="16"
        android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".U4_12_FicheiroSD"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Comprobar que o permiso está no ficheiro XML.

Binarios

Faise igual que no caso da Memoria Interna, cambiando a ruta ao ficheiro.

Por exemplo (caso de lectura):

```
File ruta = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC);
File arquivo = new File(ruta,"musica.mp3");
try {
InputStream os_interna = new FileInputStream(arquivo);
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
```

Texto

Faise igual que no caso da Memoria Interna, cambiando a ruta ao ficheiro.

Por exemplo (caso de lectura):

```
File ruta = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_MUSIC);
File arquivo = new File(ruta,"musica.mp3");
try {
     FileInputStream fis_sd = new FileInputStream(arquivo);
                  InputStreamReader isreader_sd = new InputStreamReader(fis_sd);
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
```

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

Lendo o contido dos arquivos

Unha vez que temos a referencia:

- Á clase InputStreamReader para datos de texto.
- Á clase InputStream para datos binarios.

Podemos ler o contido e traballar con el ou ben levalo a outro destino diferente (por exemplo, na sección das bases de datos imos copiar o base de datos (binario) dende o cartafol /assets/ ao cartafol /data/data/package_name/databases/)

A forma de ler / escribir é mediante un Buffer. Neste exemplo imos crear unha variable de tipo String e ir engadindo anacos de arquivo ata que rematemos de ler. Cada vez que lemos, gardamos nunha variable o número de bytes lidos, xa que na última lectura o número de bytes lidos pode ser menor que o tamaño do buffer.

O tipo de buffer dependerá do que vaiamos a ler. No caso de texto usaremos un buffer de tipo char, pero no caso de imaxes, bases de datos,...usaremos un de tipo byte.

InputStream / OutputStream <----------> byte
InputStreamReader / OutputStreamReader <----------> char

O proceso será o mesmo:

- Definimos o tamaño do buffer en char / bytes e a variable para gardar os char / bytes lidos:

```
int tamRead;
char[] buffer = new char[2048];
```

- Agora temos que ler. Usaremos o método read(buffer) da clase InputStreamReader (no caso de datos de tipo texto).

  Dito método devolve o número de caracteres lidos (bytes) e o gardaremos en tamRead.
  Isto o repetiremos ata que o método read devolva -1, co que chegaríamos ao final do ficheiro.
  A chamada ao método read pode provocar unha excepción de entrada / saída (IOException) polo que o poñemos o código entre un **try....catch(IOException ioe)** ou lanzamos un **throw IOException** dende o procedemento onde estea definido.

```
while((tamRead = isreader_raw.read(buffer))>0){

}
```

- Agora en cada iteración do bucle temos que engadir á variable String o lido.

  Para isto usaremos o método da clase String:

  copyValueOf(char[] buffer, int offset, int tamaño)

```
while((tamread = isreader_raw.read(buffer))>0){
cadea_lida += String.copyValueOf(buffer, 0, tamread);
}
```

- Por último, xa fora do bucle, pecharemos o InputStreamReader e o InputStream:

```
isreader_raw.close();
is_raw.close();
```

Nota: No caso de escribir necesitamos facer un flush, é dicir, asegurarnos que os últimos datos foron enviados a disco antes de pechar:

```
osw_sdcard.flush();
osw_sdcard.close();
```

Sendo osw_sdcard un obxecto da clase OutputStreamWriter.

O código completo dun exemplo de lectura de datos de tipo cadea que se atopan en /res/raw:

```
private void lerDatos() throws IOException{
InputStream is_raw = getResources().openRawResource(R.raw.datos);
InputStreamReader isreader_raw = new InputStreamReader(is_raw);

int tamread;
char[] buffer = new char[2048];
String cadea_lida ="";

while((tamread = isreader_raw.read(buffer))>0){
cadea_lida += String.copyValueOf(buffer, 0, tamread);
}
isreader_raw.close();
is_raw.close();
Log.i(TAG_ARQUIVOS,cadea_lida);
}
```

Nota: Dependendo da ferramenta e o entorno onde credes o arquivo txt, pode ser necesario abrir o arquivo indicando o xogo de caracteres utilizado. Así:

```
InputStreamReader isreader_raw = new InputStreamReader(is_raw,"UTF-8");
```

Estariamos lendo un arquivo co xogo de caracteres UTF-8.

Aclaración: Como comentamos inicialmente o número de clases para facer operacións e E/S é moi amplo. No caso dos arquivos de texto pódese ler liña a liña utilizando outro tipo de Clases:

```
InputStream is_raw = getResources().openRawResource(R.raw.proba);
DataInputStream dis = new DataInputStream(is_raw);
String lin = dis.readLine();
```

Escribindo contido nos arquivos

No proceso contrario, xa comentamos anteriormente que necesitamos un obxecto da clase OutputStream (datos binarios) ou OutputStreamWriter (datos de texto).

Xa sabemos conseguir dita referencia (dados nos puntos anteriores).

Unha vez o temos deberemos de chamar ao método write levando como parámetros os mesmos datos que no caso da lectura:

```
osw.write (buffer, offset, tamread)
```

Sendo osw un obxecto da clase OutputStreamWriter.

Se o contido se obtén dun control (EditText) ou o temos localmente gardado (nunha variable) podemos gardar o seu contido desta outra forma:

```
String textoGardar="Exemplo de texto,.....";
osw.write(textoGardar + "\n");
..........
osw.flush();
osw.close();
```

Así. no exemplo anterior, se queremos copiar o contido do arquivo que se atopa en /res/raw a un cartafol da tarxeta sd externa de nome /DATOS/ (previamente creado) sendo o nome do arquivo de destino "datoscopiados.txt":

```
private void lerEscribir() throws IOException{
InputStream is_raw = getResources().openRawResource(R.raw.datos);
InputStreamReader isreader_raw = new InputStreamReader(is_raw);

File fsaida = new File(Environment.getExternalStorageDirectory(),"DATOS"+File.separator+"datoscopiados.txt");
if (fsaida.exists()) fsaida.delete();

FileOutputStream fos_sdcard = new FileOutputStream(fsaida);
OutputStreamWriter osw_sdcard = new OutputStreamWriter(fos_sdcard);

int tamread;
char[] buffer = new char[2048];
String cadea_lida ="";

while((tamread = isreader_raw.read(buffer))>0){
osw_sdcard.write(buffer, 0, tamread);
cadea_lida += String.copyValueOf(buffer, 0, tamread);
}
isreader_raw.close();
is_raw.close();
osw_sdcard.flush();
osw_sdcard.close();
Log.i(TAG_ARQUIVOS,cadea_lida);

}
```

Caso Práctico

- Nome do proxecto e da activity: **UD1_03_DatosPersistentes_Arquivos**

O obxectivo desta práctica e ver, gardar e recuperar información dende diferentes lugares físicos.

A práctica consistirá na seguinte Activity:

Consta de:

- EditText multilínea na parte superior onde se vai cargar o contido dun arquivo de texto.
- Un botón 'Cargar Texto': Accede ao cartafol /res/raw/ e le o arquivo de texto subido previamente, pasando o seu contido ao EditText.
- Un botón 'Gardar Texto': Garda o contido do EditText á tarxeta SD Externa, ao cartafol Download da mesma.

- Un botón 'Copiar Imaxe': Copia unha imaxe (subida previamente) dende o cartafol /assets/ ao cartafol Download da tarxeta SD Externa.
- Un botón 'Cargar Imaxe': Carga a imaxe copiada do botón anterior (a que se atopa na SD Externa) ao ImageView.
- Un ImageView: Onde se visualiza a imaxe copiada da tarxeta SD Externa.

- Activity de manexo de arquivos
- 

Imaxe inicial da Activity.
- 

Despois de premer o botón de 'Cargar Texto', modificar o texto e premer o botón de 'Gardar Texto'.

- 

Despois de premer o botón de 'Copiar Imaxe'.

- 

Despois de premer o botón de 'Cargar Imaxe'.

- 

Arquivos copiados na SD Externa.

Preparación

- Media:pdmavanzado_badlogic.jpg: Gardar este arquivo no cartafol /assets/ do proxecto Android. Gardalo co seu nome en minúsculas.
- Media:pdmavanzadotexto.txt: Gardar este arquivo no cartafol /res/raw/ do proxecto Android. Gardalo co seu nome en minúsculas.
- Engade no AndroidManifiest.xml o permiso de escritura na SD Externa.

Creamos a Activity

- Nome do proxecto: **UD1_03_DatosPersistentes_Arquivos**
- Nome da activity: **UD1_03_DatosPersistentes_Arquivos.java**

Código do layout xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="${relativePackage}.${activityClass}" >

    <EditText
        android:id="@+id/UD1_03_txtTexto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:lines="3"
        android:textSize="12sp"
        android:inputType="textMultiLine"
        android:layout_centerHorizontal="true"
      >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/UD1_03_btnCargarTexto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/UD1_03_txtTexto"
        android:layout_centerHorizontal="true"
        android:text="CARGAR TEXTO" />

    <Button
        android:id="@+id/UD1_03_btnGardarTexto"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/UD1_03_btnCargarTexto"
        android:layout_centerHorizontal="true"
        android:text="GARDAR TEXTO" />

    <ImageView
        android:id="@+id/UD1_03_imgVwImaxe"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:src="@drawable/ic_launcher" />

    <Button
        android:id="@+id/UD1_03_btnCargarImaxe"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/UD1_03_imgVwImaxe"
        android:layout_centerHorizontal="true"
        android:text="CARGAR IMAXE" />

    <Button
        android:id="@+id/UD1_03_btnCopiarImaxe"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/UD1_03_btnCargarImaxe"
        android:layout_centerHorizontal="true"
        android:text="COPIAR IMAXE" />

</RelativeLayout>
```

Código da clase UD1_03_DatosPersistentes_Arquivos

Obxectivo: Ver, gardar e recuperar información dende diferentes lugares físicos

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;

import android.app.Activity;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Toast;

public class UD1_03_DatosPersistentes_Arquivos extends Activity {

private void xestionarEventos(){

Button btnCargarArquivoTexto = (Button)findViewById(R.id.UD1_03_btnCargarTexto);
btnCargarArquivoTexto.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub

EditText edit = (EditText)findViewById(R.id.UD1_03_txtTexto);

    InputStream is_raw = getResources().openRawResource(R.raw.pdmavanzadotexto);
    InputStreamReader isreader_raw;
try {
isreader_raw = new InputStreamReader(is_raw,"UTF-8");
} catch (UnsupportedEncodingException e1) {
// TODO Auto-generated catch block
e1.printStackTrace();
return;
}

    int tamread;
    char[] buffer = new char[2048];
    String cadea_lida ="";

    try {
while((tamread = isreader_raw.read(buffer))>0){
           cadea_lida += String.copyValueOf(buffer, 0, tamread);
}
    isreader_raw.close();
    is_raw.close();
} catch (IOException e) {
// TODO Auto-generated catch block
Toast.makeText(getApplicationContext(), "Houbo un erro na carga de datos:"+e.getMessage(), Toast.LENGTH_LONG).show();
}
    edit.setText(cadea_lida);
Toast.makeText(getApplicationContext(), "Texto cargado dende raw", Toast.LENGTH_LONG).show();

}
});

Button btnGardarArquivoTexto = (Button)findViewById(R.id.UD1_03_btnGardarTexto);
btnGardarArquivoTexto.setOnClickListener(new OnClickListener() {

@Override
```

```
public void onClick(View v) {
// TODO Auto-generated method stub
File ruta = Environment
.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File arquivo = new File(ruta, "datoscopia.txt");
EditText edit = (EditText) findViewById(R.id.UD1_03_txtTexto);

try {
FileOutputStream fos_sd = new FileOutputStream(arquivo,false);
OutputStreamWriter iswriter_sd = new OutputStreamWriter(fos_sd);
iswriter_sd.write(edit.getText().toString());
iswriter_sd.flush();

iswriter_sd.close();
fos_sd.close();

Toast.makeText(getApplicationContext(), "Texto gardado en " +arquivo.getAbsolutePath(), Toast.LENGTH_LONG).show();

} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

}
});
Button btnCopiarImaxe = (Button)findViewById(R.id.UD1_03_btnCopiarImaxe);
btnCopiarImaxe.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub

try {
File ruta = Environment
.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File arquivo_destino = new File(ruta, "badlogic_copia.jpg");
OutputStream os = new FileOutputStream(arquivo_destino);
InputStream is = getAssets().open("pdmavanzado_badlogic.jpg");
int tamRead;
byte[] buffer = new byte[2048];
while((tamRead = is.read(buffer))>0){
os.write(buffer, 0, tamRead);
                }
os.flush();
os.close();
is.close();

Toast.makeText(getApplicationContext(), "Imaxe copiada a " +arquivo_destino.getAbsolutePath(), Toast.LENGTH_LONG).show();
} catch (IOException e) {
// TODO Auto-generated catch block
Toast.makeText(getApplicationContext(), "Houbo un erro na copia da imaxe:"+e.getMessage(), Toast.LENGTH_LONG).show();
}



}
});
Button btnCargarImaxe = (Button)findViewById(R.id.UD1_03_btnCargarImaxe);
btnCargarImaxe.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub
File ruta = Environment
.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File arquivo_destino = new File(ruta, "badlogic_copia.jpg");

if (!arquivo_destino.exists()){
Toast.makeText(getApplicationContext(), "O arquivo de imaxe non existe na tarxeta SD. Tes que copialo primeiro!!!", Toast.LENGTH_LONG
return;
}
```

```
ImageView img = (ImageView)findViewById(R.id.UD1_03_imgVwImaxe);
try {

InputStream is = new FileInputStream(arquivo_destino);

img.setImageBitmap(BitmapFactory.decodeStream(is));
} catch (FileNotFoundException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}

}
});

}


@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_ud1_03__datos_persistentes__arquivos);

xestionarEventos();
}
}
```

- **Liñas 34-63**: Xestionamos o evento de Click sobre o botón Cargar Texto. Pasamos o contido do arquivo /res/raw/pdmavanzadotexto a unha variable de tipo String e pasamos o contido ao EditText.
- **Liñas 72-94**: Xestionamos o evento de Click sobre o botón Gardar Texto. Pasamos o contido do EditText a un arquivo na SD, ao cartafol Downloads, de nome "datoscopia.txt".
- **Liñas 105-125**: Xestionamos o evento de Click sobre o botón Copiar Imaxe. Pasamos a imaxe de nome pdmavanzado_badlogic.jpg ao cartafol Downloads, có nome "badlogic_copia.jpg".
- **Liñas 135-153**: Xestionamos o evento de Click sobre o botón Cargar Imaxe. Cargamos a imaxe do cartafol Downloads, có nome "badlogic_copia.jpg" ao ImageView (visto no curso de iniciación ImageView).

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2014).