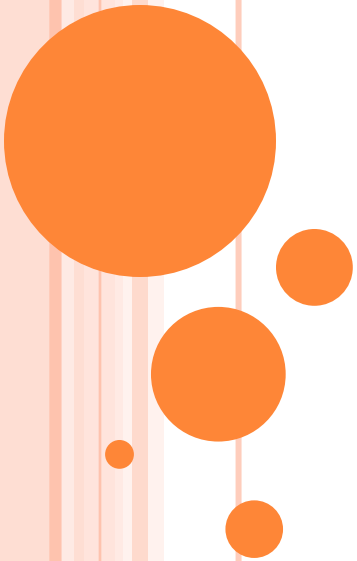


Archivos en Android

Acceso a datos

Archivos

- El espacio para guardar archivos en Android se divide en dos áreas: **área interna** y **área externa**.
- Los archivos del área interna son exclusivos de la aplicación que los crea.
- Los archivos del área externa son accesibles a todas las aplicaciones.

Archivos: permisos de lectura

- Para poder leer archivos del área externa se debe proporcionar permiso en el manifiesto.

  <uses-permission android:name= "android.permission.READ_EXTERNAL_STORAGE" />

- No es obligatorio pero sí recomendable. A partir de la versión 4 puede ser obligatorio, según la configuración del usuario.

Archivos: permiso de escritura

- Para poder escribir archivos en el área externa se debe proporcionar permiso en el manifiesto, incluye el permiso de lectura.

  <uses-permission android:name= "android.permission.WRITE_EXTERNAL_STORAGE" />

- Es obligatorio.

Archivos en Android

- Se debe considerar que la disponibilidad de espacio siempre es limitada.
- Antes de escribir archivos se debe comprobar la disponibilidad de espacio, no se debe superar el 90% del espacio disponible.
- Los métodos **getFreeSpace()** y **getTotalSpace()** de la clase File proporcionan esta información.

Archivos externos

- Los archivos externos no siempre están disponibles, de hecho, pueden estar en una tarjeta SD que es posible que esté insertada o no.
- Cada vez más dispositivos Android disponen de un área de memoria externa que no está en una tarjeta SD. Aunque además puede haber una tarjeta SD.

Archivos internos

- Los archivos de la memoria interna se guardan en **/data/data/nombre.del.paquete**.
- En esta carpeta hay dos carpetas: cache y files.
- La carpeta cache se usa como almacén temporal.
- La carpeta files se usa para guardar archivos privados de la aplicación.
- Puede haber más carpetas.

Escribir archivos internos

- **getFilesDir()**, devuelve un File con acceso a la carpeta de archivos internos;
- **getAbsolutePath()**, devuelve un String con el nombre de la carpeta;
- File f=new File(getFilesDir(),"archivo.txt");
  FileWriter fw = new FileWriter(f);
  //FileWriter(File f,boolean append)
  fw.write(texto);
  fw.flush();
  fw.close();

Si existe, escribe al final

Garantiza que se termina de grabar los datos

Leer archivos internos

- ```
  File f = new File(getFilesDir(), "archivo.txt");
  BufferedReader br = new
                      BufferedReader(new FileReader(f));
  String linea;
  while ((linea = br.readLine()) != null) {
    texto.append(linea); //StringBuilder
    texto.append('\n');
  }
  br.close();
  ```


  ```

Leer archivos externos: permiso

- Antes de leer un archivo externo se debe comprobar si se puede leer en la memoria externa.

```
public boolean isLegible() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Escribir archivos externos: permiso

- Antes de escribir un archivo externo se debe comprobar si se puede escribir en la memoria externa.

```
public boolean isModificable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}
```

Archivos externos privados

- En la memoria externa hay un espacio exclusivo para cada aplicación: **/mnt/sdcard/Android/data/paquete.aplicacion**
- **getExternalFilesDir(null)**, devuelve la carpeta externa privada
- **getExternalFilesDir(Environment.DIRECTORY_PICTURES)**, devuelve la carpeta externa privada para las imágenes

Archivos externos públicos

- En la memoria externa suele haber espacio compartido por todas las aplicaciones.
- **Environment.getExternalStorageDirectory()**, devuelve la carpeta externa pública
- **Environment.getExternalStoragePublicDirectory (Environment.DIRECTORY_PICTURES)**, devuelve la carpeta externa pública para las imágenes

Operaciones

- Las operaciones sobre los archivos siempre se realizan del mismo modo, sean externos o internos: lectura, escritura y borrado (delete()).
- Los archivos internos y los archivos externos privados de una aplicación se borran al eliminar la aplicación.

Escribir un objeto

```
FileOutputStream escribir = new
   FileOutputStream(getFilesDir().getAbsolutePath()
   +"/agenda.obj");
ObjectOutputStream escribirObjeto = new
   ObjectOutputStream(escribir);
if (objeto instanceof Serializable) {
    escribirObjeto.writeObject(objeto);
}
escribirObjeto.flush();
escribirObjeto.close();
```

Leer un objeto

```
ObjectInputStream entrada = new
   ObjectInputStream(new
   FileInputStream(getFilesDir().getAbsolutePath() +
   "/agenda.obj"));
Object o = entrada.readObject();
objeto = (ClaseDelObjeto) o;
entrada.close();
```

Mostrar una imagen en un ImageView

```
ImageView iv = (ImageView)
   v.findViewById(R.id.imagen);
```

drawable:

Imagen guardada en la carpeta "Drawable"

```
    iv.setImageDrawable(getResources().
                        getDrawable(R.drawable.imagen));
```

archivo interno o externo:

Imagen externa, se hace referencia al archivo

```
    File f = new File(carpeta, nombreArchivo);
    iv.setImageURI(Uri.fromFile(f));
```

Shared Preferences

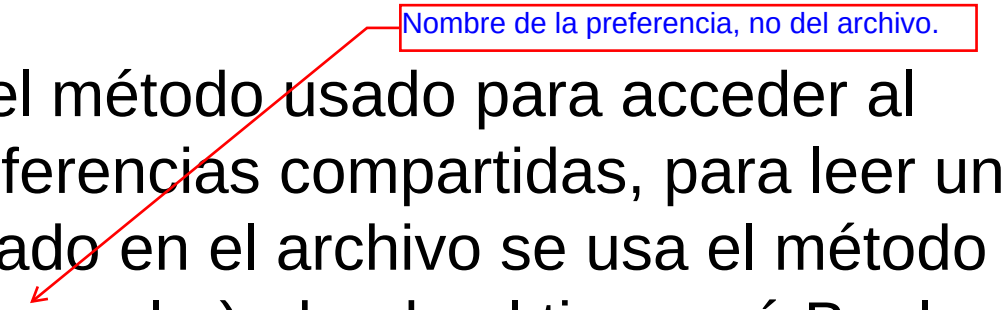
Guarda un valor aislado. Ej: color de la interfaz seleccionado por usuario. Acepta long, int, boolean, string...

- Las preferencias compartidas proporcionan una forma rápida de almacenar tipos de datos primitivos en un archivo.
- Para crear o acceder a las preferencias compartidas podemos usar uno de los siguientes métodos:
- SharedPreferences pc;
  pc = getPreferences(Context.*MODE_PRIVATE);*
  o
  pc = getSharedPreferences(nombre,
  Context.*MODE_PRIVATE);*
  o
  pc = PreferenceManager.*getDefaultSharedPreferences(*
  getApplicationContext());

Es el recomendado. Como se llama el archivo de preferencias.

Nombre de la prefencia.

Leer un preferencia compartida

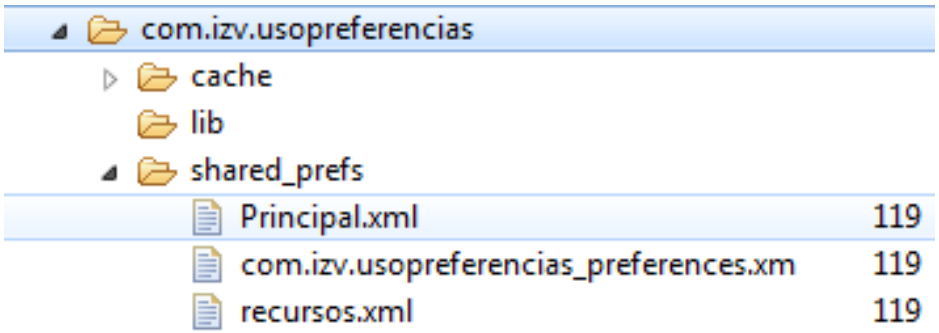


- Sea cual sea el método usado para acceder al archivo de preferencias compartidas, para leer un valor almacenado en el archivo se usa el método getTipo(nombre, valor), donde el tipo será Boolean, Float, Int, Long o String. El nombre es el nombre con el que se ha guardado el valor que se lee. Por último, valor es el valor predeterminado. Es decir, si no existe ninguna preferencia con el nombre indicado, se devuelve el valor indicado.
- String r = pc.getString("modo", "principiante");

Escribir una preferencia compartida

- Sea cual sea el método usado para acceder al archivo de preferencias compartidas, para escribir un valor almacenado en el archivo se debe obtener su editor y usar el método putTipo("nombre", valor). Para que se guarden los cambios se debe usar el método commit().
- Editor ed = pc.edit();
  ed.putString("modo", "avanzado");
  ed.commit();

Contenido de los archivos




```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
   <string name="modo">avanzado</string>
</map>
```

- http://developer.android.com/guide/topics/data/index.html
- http://developer.android.com/training/basics/data-storage/index.html