



La Clase SharedPreferences

CONCEPTOS

La clase `SharedPreferences` permite guardar datos persistentes. Cada dato se almacena con un identificador y un valor asociado a él, por ejemplo, `mes="Marzo"`, donde `mes` es la clave y `Marzo` es el valor. Los datos se guardan en archivos XML. Las siguientes constantes administran el acceso a las preferencias:

<code>MODE_PRIVATE</code>	Sólo la aplicación puede tener acceso a estas preferencias.
<code>MODE_WORLD_READABLE</code>	Las preferencias puede leerlas cualquier aplicación y sólo ésta puede modificarlas.
<code>MODE_WORLD_WRITEABLE</code>	Cualquier aplicación puede leer y modificar las preferencias.
<code>MODE_MULTI_PROCESS</code>	Comprueba la modificación de las preferencias.
<code>MODE_APPEND</code>	Agrega las nuevas preferencias con las preferencias ya existentes.
<code>MODE_ENABLE_WRITE_AHEAD_LOGGING</code>	Bandera de apertura de los datos. Si se asigna, se habilita la escritura predeterminada anticipada.

La clase `SharedPreferences` permite guardar y recuperar pares de valores clave persistentes de tipos primitivos: `boolean`, `float`, `int`, `long`, `String`. Estos datos persisten en distintas sesiones, incluso si la aplicación se cierra. También la clase `PreferenceActivity` permite crear datos preferentes del usuario.

Para tener acceso a los datos del objeto `SharedPreferences` se utilizan los siguientes métodos:

<code>getSharedPreferences()</code>	Para múltiples archivos de preferencias, identificados por su nombre, el cual se especifica con el primer parámetro.
<code>getPreferences()</code>	Para un solo archivo de preferencias de la actividad. Debido a que este será el único archivo de preferencias de la actividad, no se proporciona un nombre.

Con el método `getSharedPreferences()` se accede a las preferencias, pasándole el identificador y el acceso. Por ejemplo:

```
SharedPreferences sp = getSharedPreferences("Datos", Context.MODE_PRIVATE);
```

- Para escribir los valores:
 - a. Invocar a `edit()` para obtener un `SharedPreferences.Editor`.
 - b. Añadir los valores con los métodos `putBoolean()`, `putString()`, o algún otro, según el tipo de dato.
 - c. Guardar los nuevos valores con `commit()`.
- Para leer los valores:
 - a. Leer los datos con los métodos `getBoolean()`, `getString()`, o el que corresponda, según el tipo de dato.

Android permite otras opciones para almacenar datos persistentes, la elección depende de las necesidades de la aplicación.

- I. **Almacén interno.** En el almacén interno se almacenan datos privados en la memoria interna del dispositivo. Por definición, los archivos guardados en la memoria interna son privados para la aplicación y otras aplicaciones no pueden acceder a ellos (tampoco el usuario). Cuando el usuario desinstala la aplicación, estos archivos se eliminan.

Para crear y escribir un archivo privado al almacenamiento interno:

- i. Invocar a `openFileOutput()` con el nombre del archivo y el modo de acceso. Esto devuelve una `FileOutputStream`.
- ii. Escribir en el archivo con `write()`.
- iii. Cerrar el flujo con `close()`.

Por ejemplo:

```
String FILENAME = "misdatos";
String s = "ESCOM";
```



```
FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

Con `MODE_PRIVATE` se crea el archivo (o se sustituye en archivo con el mismo nombre) y lo convierte en privado para la aplicación. Otros modos son `MODE_APPEND`, `MODE_WORLD_READABLE` y `MODE_WORLD_WRITEABLE` (estos dos últimos son obsoletos).

Para leer un archivo del almacenamiento interno:

- i. Invocar a `openFileInput()` y pasarle el nombre del archivo a leer. Regresa un `FileInputStream`.
- ii. Leer los bytes del archivo con `read()`.
- iii. Enseguida, cerrar el flujo con `close()`.

Si se necesita guardar datos sin persistencia, se utiliza `getCacheDir()` para abrir un archivo que represente la carpeta interna en la que la aplicación guardaría los archivos cache temporales.

Otros métodos útiles son:

```
getFilesDir()    Obtiene la ruta absoluta de la carpeta en la que se guardaron los archivos.
getDir()         Crea (o abre, si existe) la carpeta en el espacio del almacén interno.
deleteFile()     Borra un archivo en el almacén interno.
fileList()       Regresa un arreglo de archivos guardados por la aplicación.
```

- II. **Almacén externo.** Es un medio extraíble (una tarjeta SD) o una memoria interna (no extraíble). Los archivos guardados en el almacenamiento externo son legibles por el mundo y puede modificarlos el usuario cuando se permite el almacenamiento masivo USB para transferir archivos en una computadora.

Para acceder al almacén externo se necesitan los permisos del sistema con `READ_EXTERNAL_STORAGE` or `WRITE_EXTERNAL_STORAGE`. Por ejemplo, en el archivo `AndroidManifest.xml`:

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

Se debe verificar si existe almacén suficiente para guardar archivos con `getExternalStorageState()`. Por ejemplo:

```
public boolean isExternalStorageWritable() { // Verifica almacén para read y write.
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

public boolean isExternalStorageReadable() { // Verifica almacén al menos para read
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

Por ejemplo, para crear una nueva carpeta para imágenes:

```
public File getAlbumStorageDir(String albumName) {
    File file = new File(Environment.getExternalStoragePublicDirectory(
```



```
        Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

DESARROLLO

EJEMPLO 1.

En el siguiente ejemplo se emplea la clase `NotificationCompat` para construir la notificación.

Paso 1. Crear un nuevo proyecto **Preferencias**. En la carpeta `java/com.example.mipaquete`, abrir y modificar el archivo `MainActivity.java` con el siguiente código:

```
import android.content.Context;
import android.content.SharedPreferences;
import android.os.*;
import android.app.*;
import android.widget.*;


public class MainActivity extends Activity{
    SharedPreferences sp;
    EditText          jetn, jetx, jety;
    String            s;
    Float             x, y;
    public void onCreate(Bundle b){
        super.onCreate(b);
        setContentView(R.layout.activity_main);
        jetn  = (EditText) findViewById(R.id.xetn);
        jetx  = (EditText) findViewById(R.id.xetx);
        jety  = (EditText) findViewById(R.id.xety);
        sp    = getSharedPreferences("preferencias", Context.MODE_PRIVATE);
        s = sp.getString("titulo", "ESCOM");
        x = sp.getFloat("x", 0);
        y = sp.getFloat("y", 0);
        jetn.setText(s);
        jetx.setText("" + x);
        jety.setText("" + y);
    }
    protected void onPause(){
        super.onPause();
        s = jetn.getText().toString();
        x = Float.parseFloat(jetx.getText().toString());
        y = Float.parseFloat(jety.getText().toString());
        SharedPreferences.Editor miEditor = sp.edit();
        miEditor.putString("titulo", s);
        miEditor.putFloat("x", x);
        miEditor.putFloat("y", y);
        miEditor.commit();
        Toast.makeText(this, "Preferencias guardadas", Toast.LENGTH_LONG).show();
    }
}
```

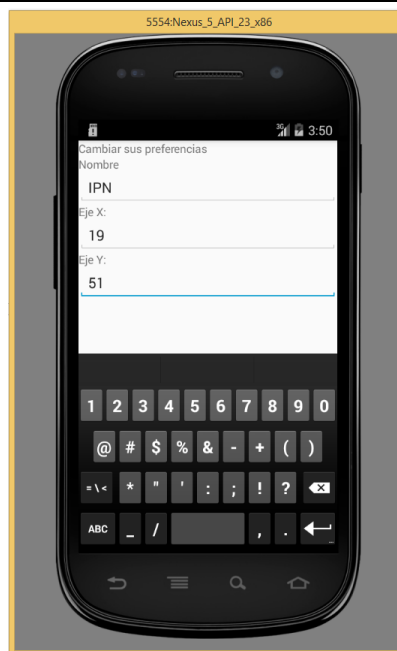
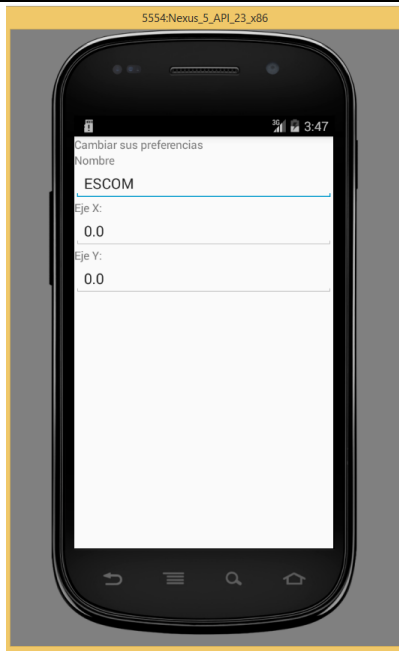
Paso 2. En la carpeta `res/layout`, abrir y modificar el archivo `activity_main.xml` con el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
```

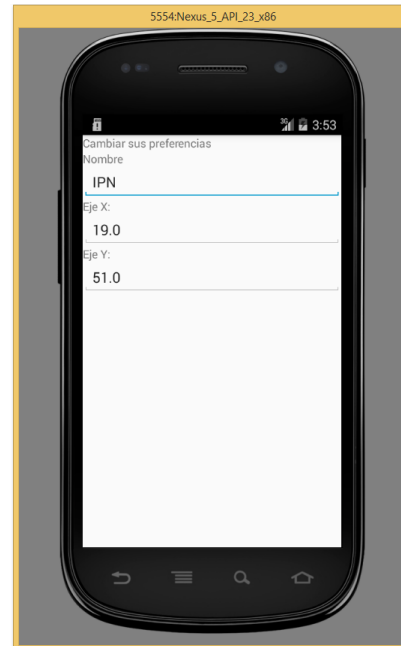
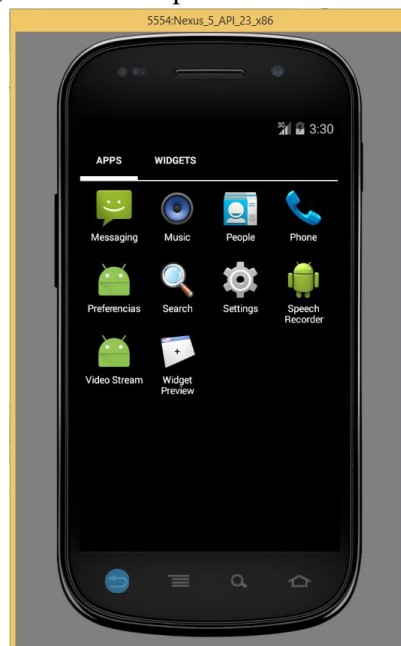



```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:id="@+id/layout1" >
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Cambiar sus preferencias"
    android:id="@+id/text1" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Nombre" />
<EditText
    android:id="@+id/xetn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="EditText" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Eje X:" />
<EditText
    android:id="@+id/xetx"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="EditText" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Eje Y:" />
<EditText
    android:id="@+id/xety"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="EditText" />
</LinearLayout>
```

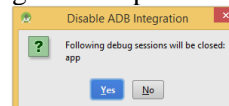
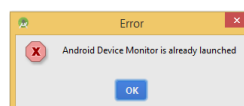
Paso 4. Ahora, ejecutar la aplicación. Cuando la aplicación se ejecuta por primera vez se muestra la inicialización de los datos de las preferencias. Si el usuario ingresa nuevos datos y la aplicación se cierra, digitando el botón  de la barra inferior, los datos se guardan, indicándose ello con el mensaje **Preferencias guardadas** del Toast y los datos permanecerán persistentes.



Al reabrir la aplicación, se muestra la persistencia con los mismos datos:



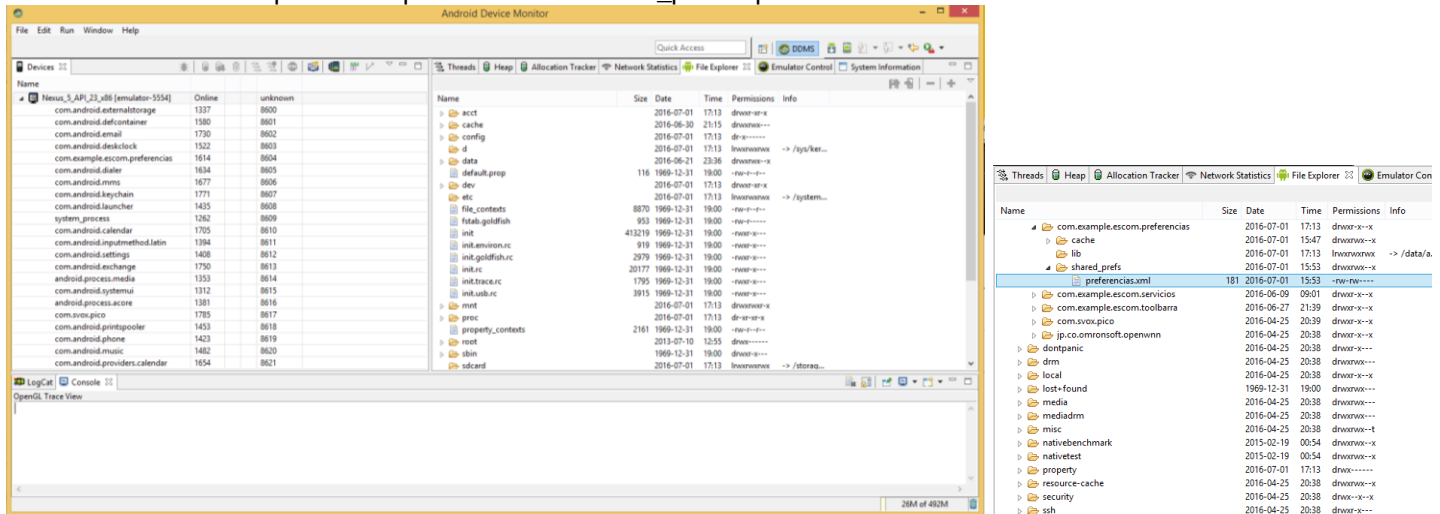
Paso 5. En el menú superior del IDE, digitar el botón del icono ADM Android Device Monitor  para mostrar su ventana; allí se busca el archivo XML de las preferencias. La ventana del ADM puede tardar varios segundos en mostrarse; si se insiste en mostrar la ventana se muestra un mensaje de error; digitar OK. Si la aplicación se encuentra en ejecución, se muestra un mensaje que solicita la terminación; digitar **Yes** para cerrar la aplicación y esperar:



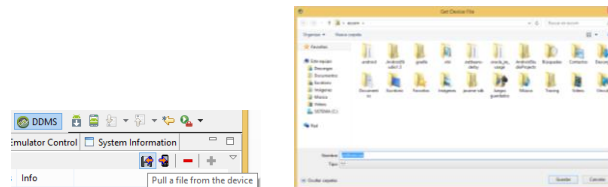


Paso 6. Ahora, después de varios segundos, se muestra la ventana ADM. En la sección derecha, seleccionar la pestaña **File Explorer** y en el árbol de directorios seleccionar lo siguiente:
data->data->com.example.mipaquete->shared_prefs->datos_de_preferencias.xml

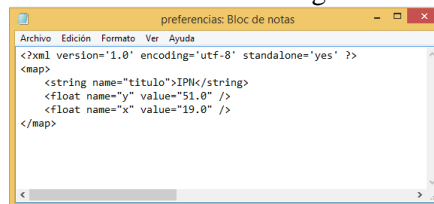
En este caso, el paquete es com.example.escom.preferencias y el archivo es preferencias.xml, es decir:
data->data->com.example.escom.preferencias->shared_prefs->preferencias.xml



Paso 7. Para extraer el archivo preferencias.xml, digitar el botón **Pull a file from the device**. Se muestra la ventana de diálogo de archivos:



Guardar y abrir el archivo preferencias.xml. Se muestran los datos guardados, como se indica en la siguiente imagen:



EJERCICIO 1. Diseñar una aplicación que solicite el nombre y apellidos del usuario para almacenarlos en un almacén interno del dispositivo. Después, mostrar en un TextView los datos almacenados a través de un Intent.

EJERCICIO 2. Diseñar una aplicación para almacenar una imagen, seleccionada desde la galería de imágenes, en un almacén externo del dispositivo. Después, mostrar en un Canvas la imagen almacenada a través de un Intent.