PERSISTENCIA DE DATOS

Pág. 165 y sgts. del Manual de SGOliver

1. INTRODUCCION

- En Android existen tres formas de almacenar información de forma permanente:
  - Mediante lo que se llaman "**preferencias compartidas**", lo que permite almacenar conjuntos sencillos de datos en forma de pares clave/valor.
  - Sistemas tradicionales de archivos.
  - **Sistema de base de datos relacional (base de datos SQLite**).

2. PREFERENCIAS COMPARTIDAS

Más información en https://developer.android.com/guide/topics/data/data-storage.html#pref

- Android proporciona la clase ***SharedPreferences*** que permite guardar datos sencillos de una forma simple.
- Los datos se guardan en **parejas *clave/valor***, es decir, cada dato estará compuesto por un **identificador** único y un **valor** asociado a dicho identificador, de forma similar a como se hacía con los objetos de tipo ***Bundle***, salvo que en este caso se almacenan de forma permanente.
- Podemos crear múltiples objetos ***SharedPreferences*** pero disponemos de uno por defecto al que accederemos mediante el método ***getDefaultSharedPreferences()*** de la clase ***PreferenceManager.*** Este método recibe como parámetro el contexto de la actividad.

`static SharedPreferences`	`getDefaultSharedPreferences(Context context)` Gets a SharedPreferences instance that points to the default file that is used by the preference framework in the given context.

```
SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(MainActivity.this);
```

- No podemos modificar el contenido del objeto ***SharedPreferences*** directamente sino que tenemos que asociar dicho objeto a un editor de tipo ***SharedPreferences.Editor***, mediante el método ***edit()***.

abstract SharedPreferences.Editor	edit() Create a new Editor for these preferences, through which you can make modifications to the data in the preferences and atomically commit those changes back to the SharedPreferences object.

```
SharedPreferences.Editor editor = prefs.edit();
```

- Una vez obtenida la referencia al editor, utilizaremos los métodos ***put()*** correspondientes al tipo de datos de cada preferencia (por ejemplo, ***putString()*** o ***putInt()***) para insertar o actualizar su valor.

```
editor.putString("email", "yo@email.com");
editor.putString("nombre", "Yo");
```

- Finalmente, una vez completados todos los datos necesarios, ejecutaremos el método ***apply()*** para almacenarlos (también se puede usar ***commit()***).

abstract void	apply() Commit your preferences changes back from this Editor to the SharedPreferences object it is editing.
abstract SharedPreferences.Editor	clear() Mark in the editor to remove *all* values from the preferences.
abstract boolean	commit() Commit your preferences changes back from this Editor to the SharedPreferences object it is editing.

```
editor.apply();
```

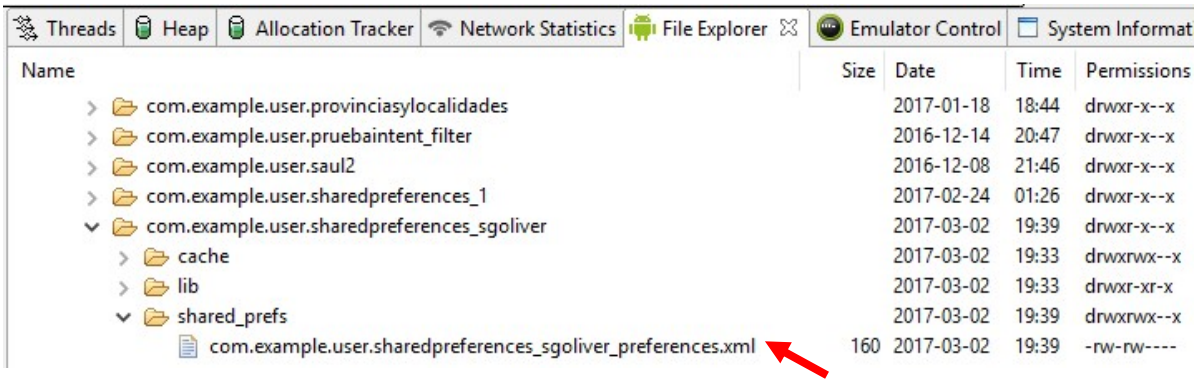
- Para leer el estado no es necesario el editor, sino que basta con usar el método ***get()*** (de la clase ***SharedPreferences***) correspondiente al tipo de dato que queramos leer, por ejemplo ***getString()***. El método ***get()*** recibe dos parámetros: la clave de la preferencia que queremos recuperar y un valor por defecto que será devuelto por el método en caso de que no se encuentre el dato con esa clave.

```
String correo=prefs.getString("email", "email_defectivo@email.com");
```

- Las preferencias compartidas no se almacenan en archivos binarios (como por ejemplo las bases de datos SQLite), sino en archivos XML. Estos ficheros XML se guardan en una ruta que sigue el siguiente patrón:

  /data/data/nombre_del_paquete/shared_prefs/nombre_preferencias.xml

  Podemos comprobarlo accediendo al explorador de ficheros con la herramienta ***Android Device Monitor***, como se muestra en la captura siguiente:




```
1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3  <string name="nombre">Luly Vázquez</string>
4  <string name="email">luly@email.com</string>
5  </map>
```

- Podemos probar esto en el **ejercicio1**.

3. PREFERENCIAS COMPARTIDAS CON LA CLASE PreferenceActivity

- Las preferencias compartidas nos pueden permitir gestionar fácilmente las opciones de una aplicación creando los objetos necesarios y añadiendo o recuperando los valores de dichas opciones a través de los métodos correspondientes (***putString()*** – ***getString()***; ***putInt()*** – ***getInt()***…).

- Sin embargo, Android dispone de una forma alternativa para definir un conjunto de opciones para una aplicación y crea por nosotros las pantallas necesarias para permitir al usuario que modifique dichas opciones.

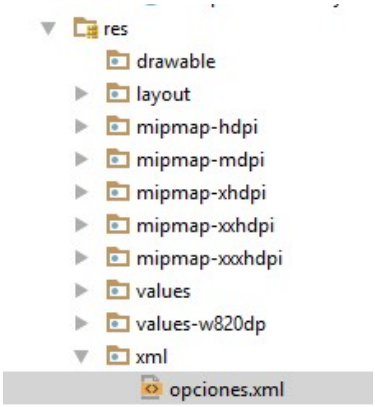
- Para ello, creamos una pantalla que va a contener las preferencias de nuestra aplicación. Esto se puede hacer usando fragmentos o, como vamos a hacer nosotros, creando una **Activity que hereda de la clase *PreferenceActivity***. Dicha Activity sobreescribe el método onCreate() incluyendo una llamada al método ***addPreferencesFromResource()***, que tiene como parámetro el fichero XML en el que hemos definido la pantalla de opciones.

```
public class OpcionesActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.opciones);
    }
}
```

- Esta actividad, al heredar de ***PreferenceActivity***, se encargará por nosotros de crear la interfaz gráfica de nuestra lista de opciones según la hayamos definido en el XML

y se preocupará por nosotros de mostrar, modificar y guardar las opciones cuando sea necesario tras la acción del usuario.

- El fichero XML que contiene la vista de nuestras preferencias debe ir en la carpeta ***res/xml***.



- En primer lugar, el elemento raíz contenedor de nuestras preferencias será ***<PreferenceScreen>***. Este elemento representará a la pantalla de opciones en sí y, a su vez, puede incorporar directamente las preferencias que queramos utilizar o podremos dividirlas en diferentes categorías. Para utilizar distintas categorías necesitamos añadir un elemento ***<PreferenceCategory>*** por cada categoría que queramos definir. Para cada categoría podemos indicar un texto descriptivo mediante el atributo ***android:title***.

- Dentro de cada categoría podemos añadir cualquier número de opciones, las cuales pueden ser de diferentes tipos:

  - **CheckBoxPreference**. CheckBox para habilitar/deshabilitar. Sus propiedades principales son:
    - **android:key**, valor interno de la preferencia.
    - **android:title**, nombre de la preferencia a mostrar.
    - **android:sumary**, breve descripción de la preferencia.

```
<CheckBoxPreference
    android:key="opcion1"
    android:title="Preferencia 1"
    android:summary="Descripción de la preferencia 1" />
```

  - **EditTextPreference**. Permite introducir valores de texto. Inicialmente muestra el nombre de la preferencia y una descripción. Al pulsar sobre ella se abre un cuadro de diálogo en el que aparece un *EditText* para que el usuario escriba el valor a almacenar. Además de los tres atributos indicados en el caso anterior, también existe **android:dialogTitle** para indicar el texto a mostrar en el cuadro de diálogo.

```
<EditTextPreference
    android:key="opcion2"
    android:title="Preferencia 2"
    android:summary="Descripción de la preferencia 2"
    android:dialogTitle="Introduce valor" />
```

  - **ListPreference**. Al pulsar sobre una opción de este tipo se mostrará la lista de valores posibles y el usuario podrá seleccionar sólo uno de ellos. Además de

los cuatro atributos comentados para los casos anteriores, para el elemento **<ListPreference>** existen dos más:

- **android:entries**, cada uno de los valores a visualizar en la lista.
- **android:entryValues**, valores internos correspondientes a cada uno de los valores de la lista que se muestra al usuario.

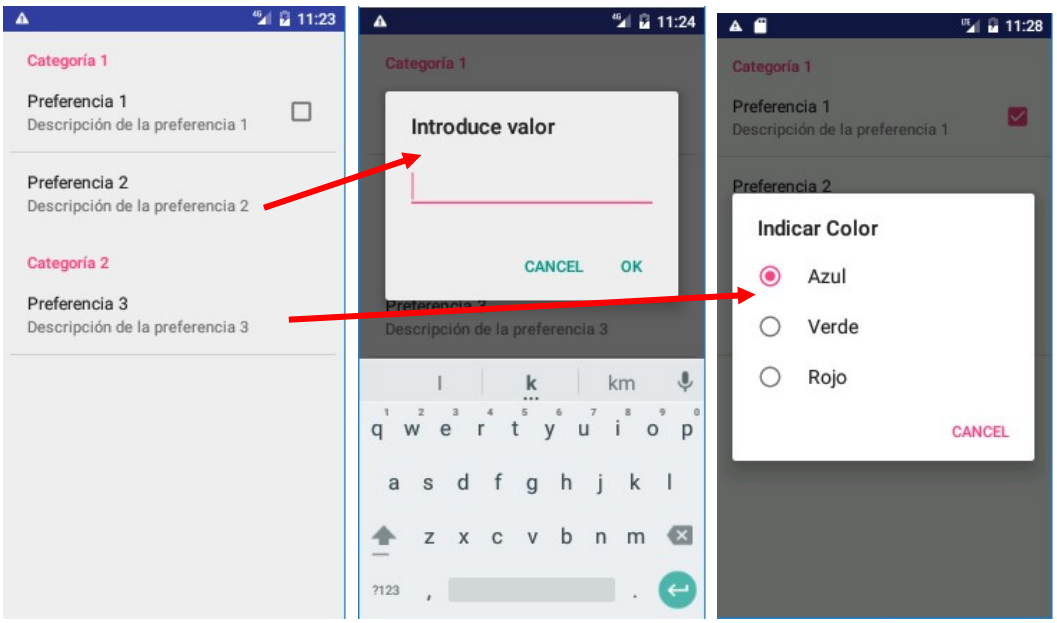
Estas listas de valores también serán ficheros XML dentro de la carpeta ***res.***

```
<ListPreference
    android:key="opcion3"
    android:title="Preferencia 3"
    android:summary="Descripción de la preferencia 3"
    android:dialogTitle="Indicar color"
    android:entries="@array/colores"
    android:entryValues="@array/codigocolores" />
```

```
<resources>
    <string-array name="colores">
        <item>Azul</item>
        <item>Verde</item>
        <item>Rojo</item>
    </string-array>
    <string-array name="codigocolores">
        <item>1</item>
        <item>2</item>
        <item>3</item>
    </string-array>
</resources>
```

- **MultiSelectListPreference**. Similar a la anterior, pero se trata de una lista de valores de selección múltiple.

- **Ejemplo:**



- Por último, habría que añadir a nuestra aplicación algún mecanismo para mostrar esta ventana de preferencias.

- Podemos probar esto como **Ejercicio 3.**