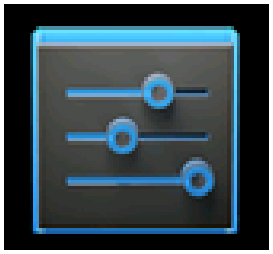
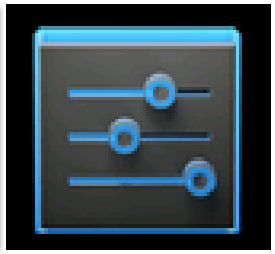


Preferencias

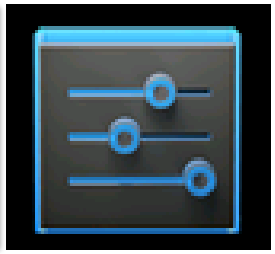
Almacenamiento de datos simples

Preferencias



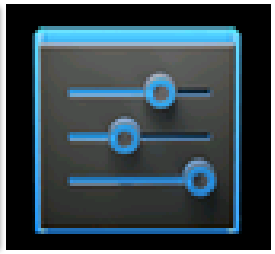
- Las preferencias no son más que **datos** que una aplicación debe guardar para personalizar la experiencia del usuario, por ejemplo información personal, opciones de presentación, etc.
- Podemos decir que “shared preferences” tiene la funcionalidad inicial de **recordar un valor y compartirlo** con otros métodos o actividades.
- Lo realmente interesante de esta funcionalidad es que este valor **continuará grabado y recordado aunque cerremos nuestra aplicación o reiniciemos el dispositivo**.

Consejos sobre preferencias de las aplicaciones



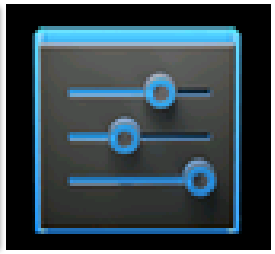
- Se considera que las buenas aplicaciones tienen una **cantidad correcta** de opciones, permitiendo a los usuarios variar el comportamiento de sus aplicaciones, sin ser demasiado complicadas
- Lo más usual, es añadir las preferencias en la primera Actividad de la aplicación como opción (menú de opciones).
- Las preferencias puede ser usadas en cualquier parte y momento de la aplicación, pueden usarse al pulsar un botón o en otros eventos. Pero lo normal, puesto que se trata de parámetros de configuración de la aplicación, es usar los eventos **onStart** y **onDestroy** de la aplicación Android para **leer** y **guardar** respectivamente en el fichero de preferencias.
- Al igual que sucede con los menús, podemos tener varios niveles de Preferencias. Los Ajustes del sistema de Android son el mejor ejemplo.

Gestión de las preferencias



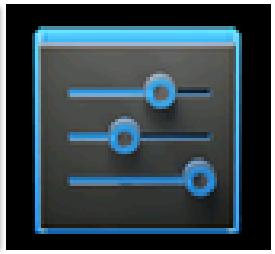
- Los datos se guardan en ficheros XML.
- Podemos gestionar las preferencias:
  - desde código Java (utilizado para guardar pocos datos)
  - desde la interfaz gráfica como un nuevo recurso de tipo xml

Gestión desde código: Clase SharedPreferences

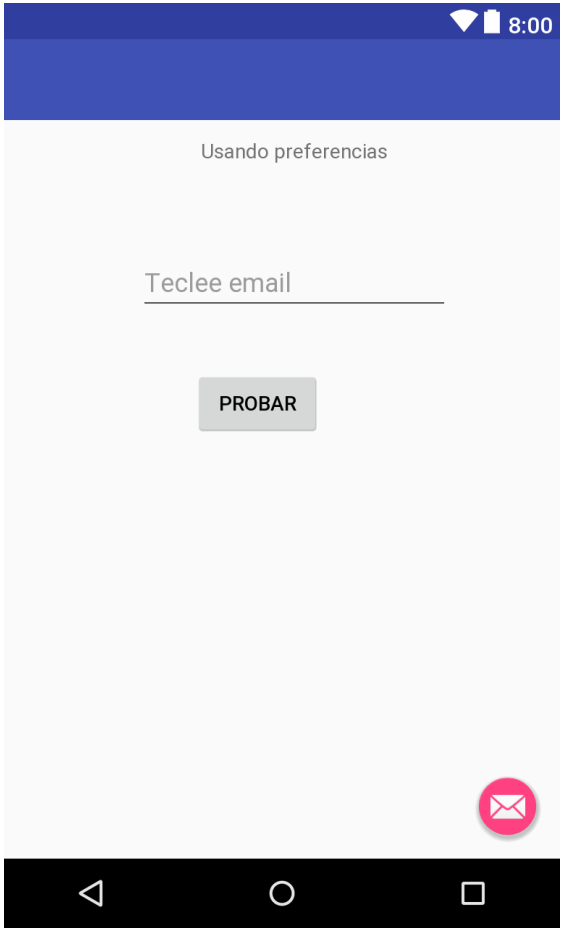


- Toda la gestión se centraliza en la clase SharedPreferences, que representará a una colección de preferencias.
- Una aplicación Android puede gestionar varias colecciones de preferencias, que se diferenciarán mediante un identificador único.
- Cada preferencia dentro de su colección se almacenará en forma de clave-valor, es decir, cada una de ellas estará compuesta por un identificador único (p.e. “email”) y un valor asociado a dicho identificador (p.e. “prueba@email.com”).

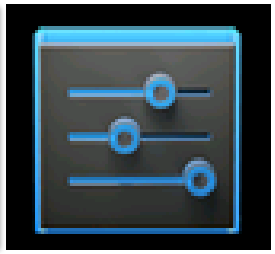
Ejemplo



```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
         …
        final EditText editText = findViewById(R.id.editText);
        final SharedPreferences prefe
         =getSharedPreferences("datos",Context.MODE_PRIVATE);
        editText.setText(prefe.getString("mail", ""));
        Button button =  findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                SharedPreferences.Editor editor = prefe.edit();
                editor.putString("mail", editText.getText().toString());
                editor.commit();
                finish();
            }
        });
```



Método getSharedPreferences()

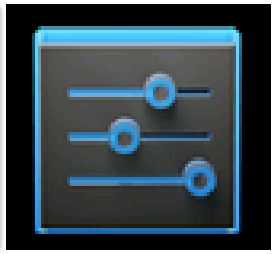


- Para obtener una referencia a una colección determinada se usa el método **getSharedPrefences()** al que se pasa el identificador de la colección y un *modo de acceso*.

  en el ej, SharedPreferences prefe=getSharedPreferences("datos",Context.MODE_PRIVATE);
- El modo de acceso indicará qué aplicaciones tendrán acceso a la colección de preferencias y qué operaciones tendrán permitido realizar sobre ellas. Hay tres posibilidades principales:
  - MODE_PRIVATE solo la aplicación puede acceder al archivo de preferencias.
  - MODE_WORLD_READABLE otras aplicaciones pueden consultar el archivo de preferencias. Obsoleto
  - MODE_WORLD_WRITEABLE otras aplicaciones pueden consultar y modificar el archivo. Obsoleto
- Una vez obtenida una referencia a la colección de preferencias, ya se puede obtener, insertar o modificar preferencias utilizando los métodos *get* o *put* correspondientes al tipo de dato de cada preferencia.
- El método **getPreferences()** es similar, se puede usar cuando solo se necesite un fichero de preferencias. No permite elegir nombre, lo asigna el sistema y es el nombre de la actividad con el sufijo xml. **RECOMENDADO SI SOLO SE UTILIZAN EN UNA ACTIVIDAD**.

  SharedPreferences prefe= getPreferences(Context.MODE_PRIVATE);

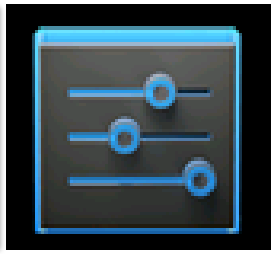
Recuperar preferencias: getString()



- Al método **getString() se pasa el nombre de la preferencia que se desea recuperar y un segundo parámetro con un valor de tipo string por defecto**.
- Este valor por defecto **será el devuelto** por el método getString() **si la preferencia solicitada no existe en la colección**.

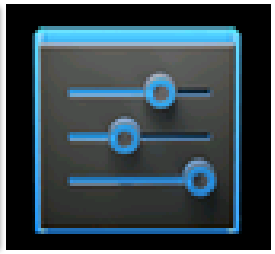
en el ejemplo, prefe.getString("mail","")

Métodos para leer preferencias



- **contains()**: comprueba si existe una preferencia en el fichero XML.
- **getAll()**: obtiene todos los valores de todas las preferencias guardadas en el fichero XML.
- **getBoolean()**: obtiene una preferencia de tipo Booleano (true, false).
- **getFloat()**: obtiene una preferencia de tipo Float (numérico).
- **getInt()**: ontiene una preferencia de tipo Int (entero).
- **getLong()**: obtiene una preferencia de tipo Long (entero largo).
- **getString()**: obtiene una preferencia de tipo String (texto).
- **getStringSet()**: obtiene una preferencia de tipo StringSet (conjunto de cadenas).
- **registerOnSharedPreferenceChangeListener()**: registra un "listener" que se invoca cuando ocurre un cambio en las preferencias.
- **unregisterOnSharedPreferenceChangeListener()**: anula el registro del "listener" anterior.

Escribir preferencias: edit() y putString()



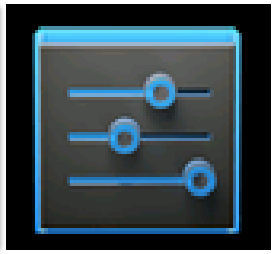
- Para actualizar o insertar nuevas preferencias el proceso será igual de sencillo, con la única diferencia de que la actualización o inserción no la haremos directamente sobre el objeto SharedPreferences, sino sobre su objeto de edición **SharedPreferences.Editor**. A este último objeto se accede mediante el método edit() de la clase SharedPreferences.

  en el ejemplo, Editor editor = prefe.edit();
- Una vez obtenida la referencia al editor, se usan los métodos *put* correspondientes al tipo de datos de cada preferencia para actualizar/insertar su valor, por ejemplo **putString(clave, valor),** para actualizar una preferencia de tipo String.

  en el ejemplo, editor.putString("mail", editText.getText().toString());
- Finalmente, una vez actualizados/insertados todos los datos necesarios llamaremos al método **commit()** para confirmar los cambios.

  en el ejemplo, editor.commit();

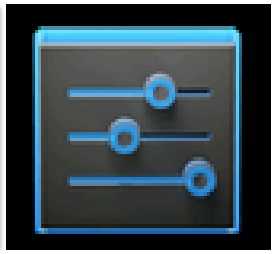
Métodos para escribir preferencias



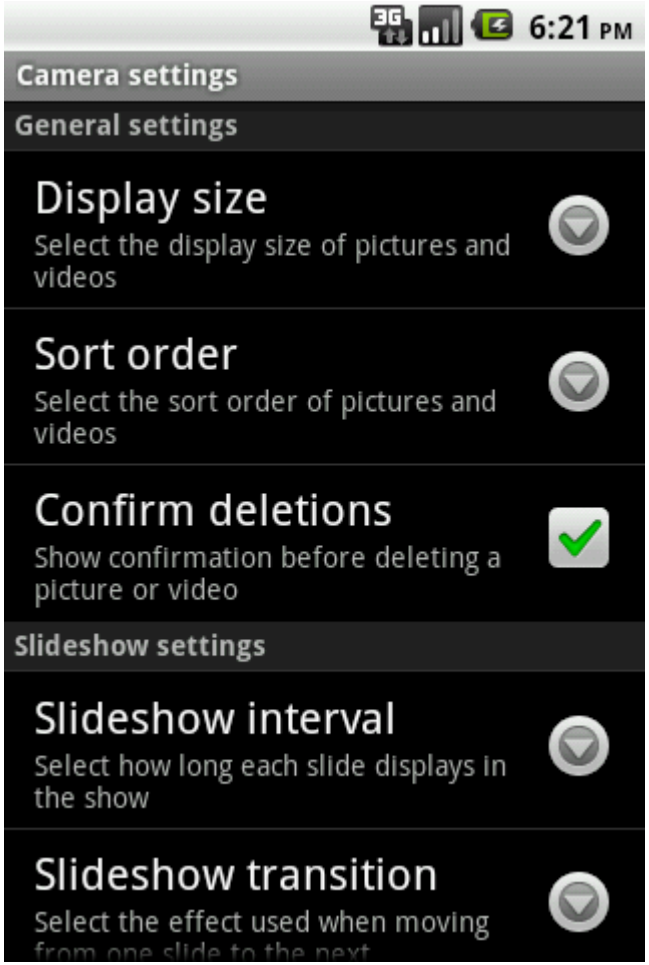
- **apply()**: guarda los cambios realizados en las preferencias cuando el objeto SharedPreferencias está en modo edición.
- **clear()**: elimina todos los valores de las preferencias.
- **commit()**: guarda los cambios realizados en las preferencias.
- **putBoolean()**: guarda un valor de tipo booleano (true, false).
- **putFloat()**: guarda un valor de tipo Float.
- **putInt()**: guarda un valor de tipo Int.
- **putLong()**: guarda un valor de tipo Long.
- **putString()**: guarda un valor de tipo String.
- **puntStringSet()**: guarda un valor de tipo StringSet.
- **remove()**: elimina el valor de una preferencia.

(Observación: mejor apply() o commit()?)

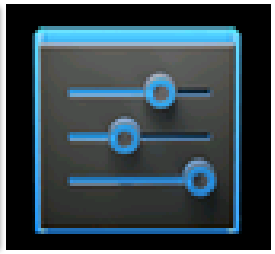
Gestión desde UI



- Android ofrece una forma alternativa de definir mediante XML un conjunto de opciones para una aplicación y crear las pantallas necesarias para permitir al usuario modificarlas a su antojo.
- Podemos hacer toda la interfaz de la pantalla de preferencias con XML, como si fuese una la interfaz de una actividad más
- Si nos fijamos en cualquier pantalla de preferencias estándar de Android veremos que todas comparten una interfaz común, similar por ejemplo a la que se muestra en la imagen contigua

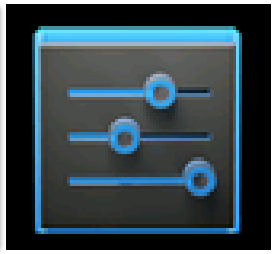


Pasos para elaborar las preferencias de la actividad



1. Añadir al fichero de construcción la biblioteca de preferencias.
2. Crear pantalla de opciones, el fichero de preferencias (un recurso de tipo xml)
3. Crear un fragmento del tipo PreferenceFragmentCompat asociado al fichero anterior
4. Crear una actividad que utilice el fragmento

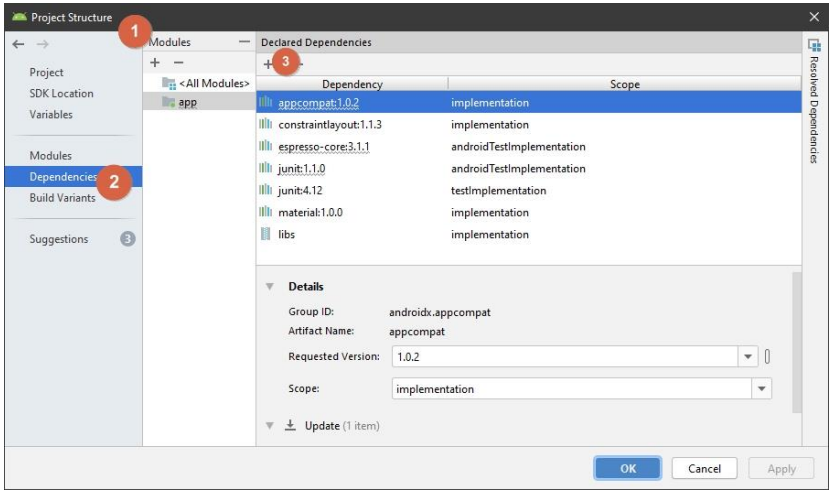
Biblioteca de Preference de AndroidX

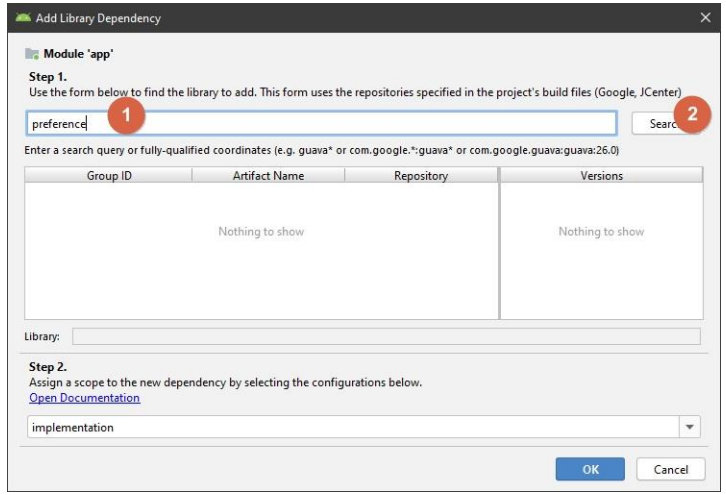


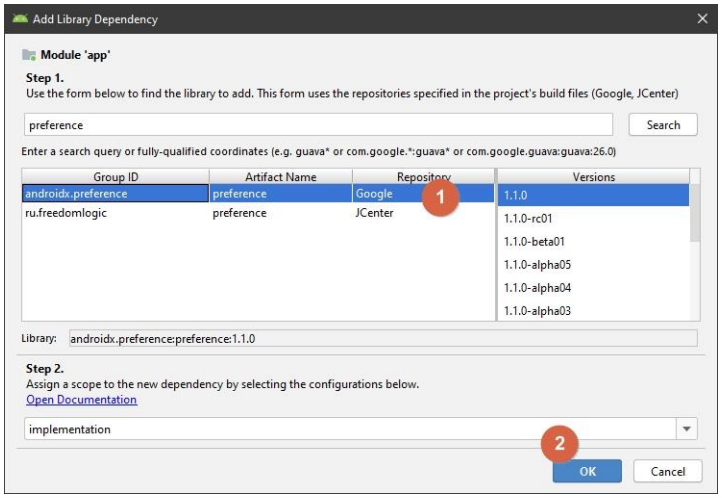
- Hay que incluir la Biblioteca de Preference de AndroidX. Esta biblioteca administra la interfaz de usuario e interactúa con el almacenamiento de modo que solo se definan las configuraciones individuales que el usuario pueda ajustar. La biblioteca incluye un tema de Material que proporciona una experiencia del usuario coherente en todos los dispositivos y todas las versiones de SO.
- Método 1:

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.0.2'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.android.material:material:1.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.0'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
    implementation 'androidx.preference:preference:1.1.0'
}
```

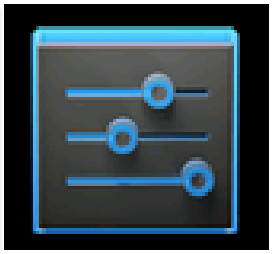
Biblioteca de Preference de AndroidX (Método 2)



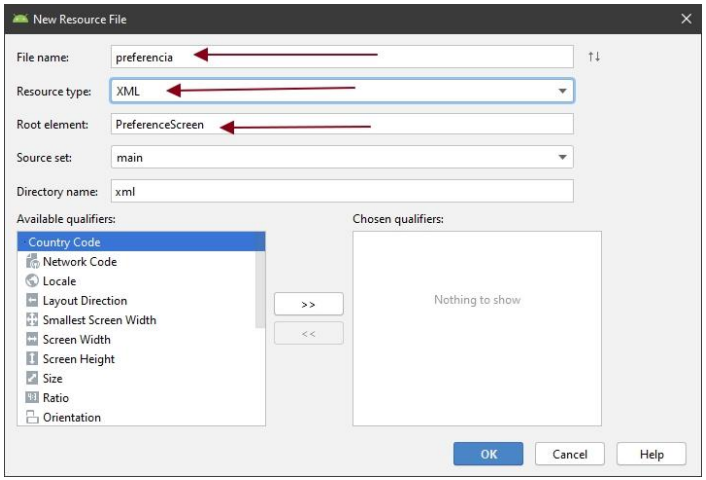




Fichero de preferencias I

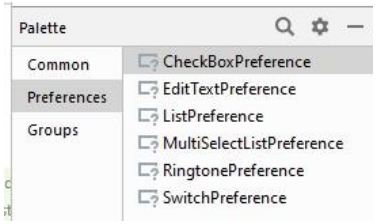


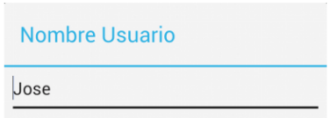
- Nuestra pantalla de opciones la vamos a definir mediante un XML (llamado por ejemplo, preferencias.xml):



- El contenedor principal de nuestra pantalla de preferencias será el elemento <PreferenceScreen>. Representa la pantalla de opciones en sí, dentro de la cual incluiremos el resto de elementos.
- Podremos incluir nuestra lista de opciones organizadas por categorías, que se representará mediante el elemento <PreferenceCategory> al que daremos un texto descriptivo utilizando su atributo android:title.

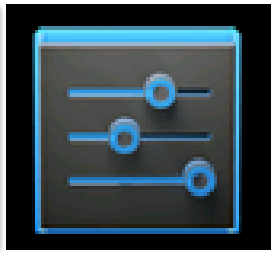
Fichero de preferencias II





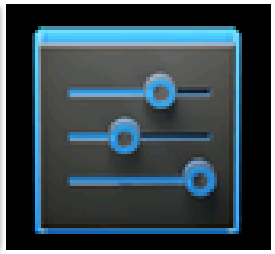
- Dentro de cada categoría podremos añadir cualquier número de opciones, las cuales pueden ser de distintos tipos (cada uno de ellos requiere la definición de diferentes atributos), entre los que destacan:
  - **CheckBoxPreference**: Muestra un texto, con un checkbox, que puede estar habilitado o no. Se guarda como booleano (true o false).
  - **EditTextPreference**: Abre un diálogo con un campo de texto. Se guarda la cadena de texto.
  - **ListPreference**: Abre un dialogo con una lista de botones radio. Se guarda el valor asignado a la opción marcada. Hay que asignarle 2 arrays, uno en la propiedad entries con las cadenas de texto a mostrar y otro en entryValues con los valores que se quiere guardar para cada cadena.
  - **MultiSelectListPreference**: Similar a la anterior pero permite selección múltiple de opciones.
- Buen enlace con ejemplos

Atributos de las opciones de preferencias



- Comunes a todos los tipos:
  - **key**: Las preferencias no tiene id, sino una llave
  - **title**: No hay un campo de texto, sino un título
  - **summary**: Un campo de descripción, que no debe ser largo
- Propios de algunos tipos:
  - **dialogTitle**: el texto a mostrar en un EditTextPreference o en ListPreference
  - **entries** y **entryValues**: necesarios en ListPreference y en MultiSelectListPreference, indican respectivamente la lista de valores a visualizar y los valores internos que utilizaremos en el código para cada uno de los valores de la lista anterior. Los valores están guardados como tipo <string-array> en el fichero res/values/strings.xml. **AUNQUE AS DIGA LO CONTRARIO ☹** Leer
  - **defaultValue**: valor inicial **AUNQUE AS DIGA LO CONTRARIO ☹**
  - **useSimpleSummaryProvider**: para fijar si el resumen de la preferencia debe mostrar el elemento guardado actualmente

Fragmento de las preferencias

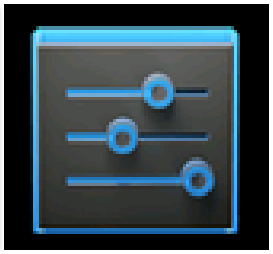


- Necesitamos crear una clase (en nuestro ejemplo llamada MisPreferencias) que asuma las preferencias creadas:

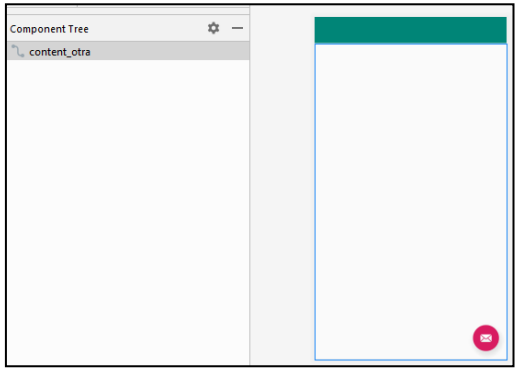
```
public class MisPreferencias extends PreferenceFragmentCompat {
    @Override
    public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
        setPreferencesFromResource(R.xml.preferencia, rootKey);
    }
}
```

Nombre que hemos dado al fichero en el paso anterior.

Actividad que guarda las preferencias



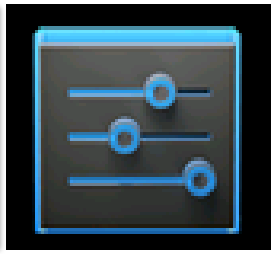
- La actividad que trabaja las preferencias tendrá un layout (vacío pero identificado) en el que mediante código se agregará el fragmento anterior:



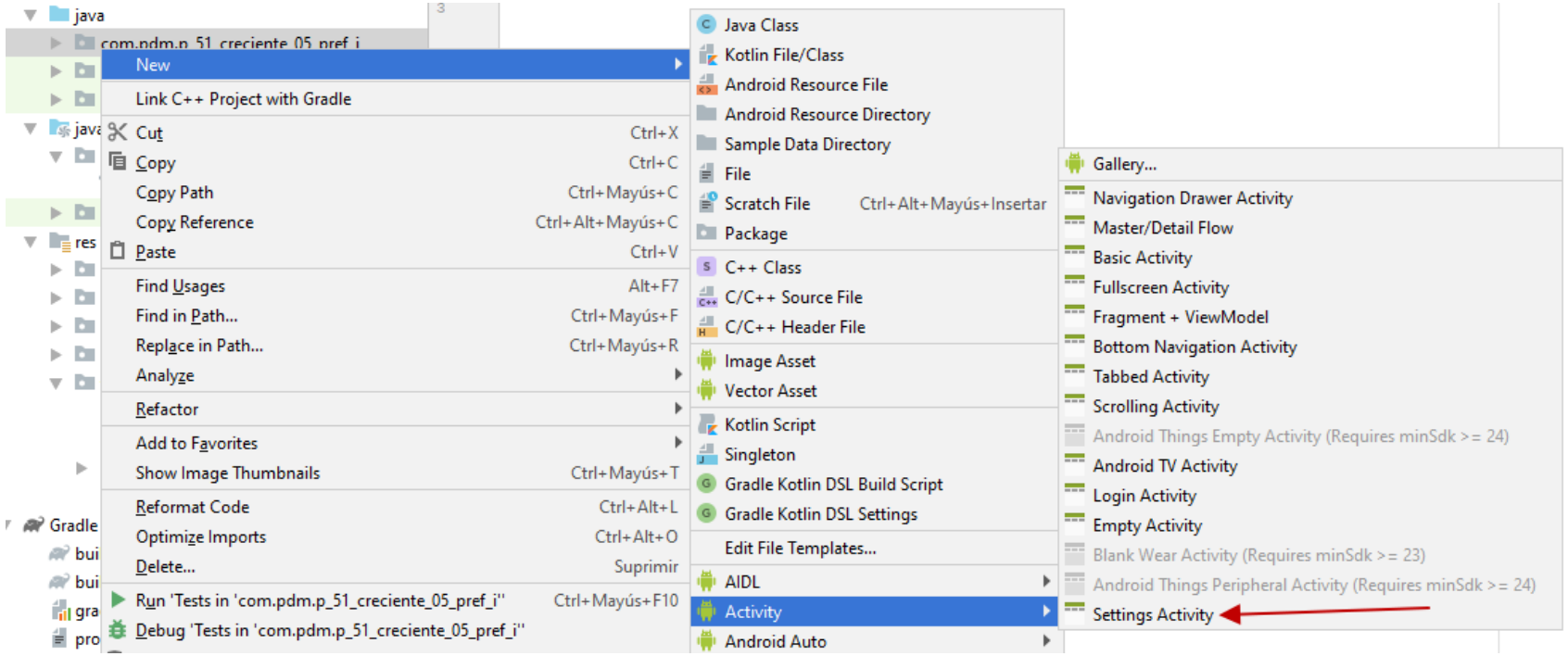

```
public class OtraActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        .....

        Fragment fragment=new MisPreferencias();
        getSupportFragmentManager().beginTransaction().replace(R.id.content_otra,fragment).commit();
    }


}
```

- A esta clase es a la que llamaremos mediante un Intent (por ejemplo, al seleccionar el ítem de un menú de opciones).

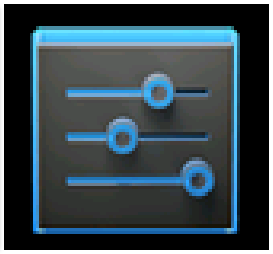
Asistente de AS



- AS nos proporciona una plantilla de actividad para hacer todas las tareas anteriores. Solo tendremos que hacer los cambios necesarios en el fichero xml de preferencias:



Actividad que recupera las preferencias guardadas



- Para recuperar el objeto SharedPreferences que se esté usando, hay que llamar al método **PreferenceManager.getDefaultSharedPreferences()**. Este método funciona desde cualquier lugar de la aplicación.

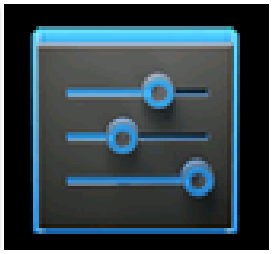
  ```
  SharedPreferences sharedPreferences =
          PreferenceManager.getDefaultSharedPreferences(getApplicationContext());
  ```

- Para recuperar los valores, se usan los mismos métodos vistos anteriormente. Por ejemplo, para leer el valor de una preferencia con *key* de nombre "email":

  ```
  String leido=sharedPreferences.getString("email", "");
  ```

- Más información sobre preferencias

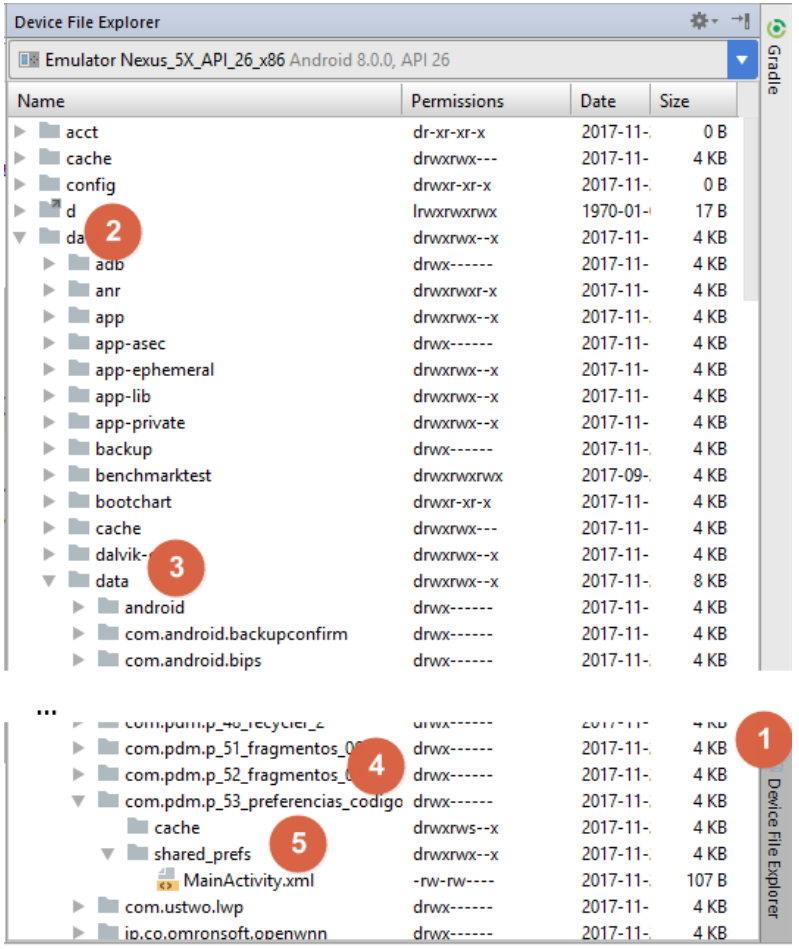
Ruta de almacenamiento de preferencias compartidas



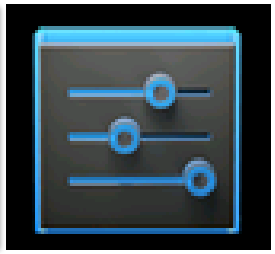
- En ficheros XML guardados en la ruta

  /data/data/paquetejava/shared_prefs/nombre_coleccion.xml

- Esta carpeta es visible desde la perspectiva Device File Explorer de AS



Prácticas propuestas



- Realiza la hoja de ejercicios Ejercicios_12_Preferencias