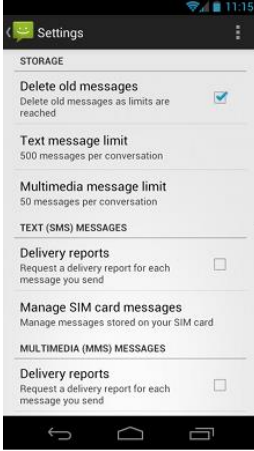


¿CÓMO USAR LAS PREFERENCIAS PARA GESTIONAR UNA PANTALLA DE CONFIGURACIÓN?

CREAR LA PANTALLA DE CONFIGURACIÓN

1. **Crear la pantalla de configuración** usando un archivo XML que se guarda en res/xml. Por ejemplo, para esta pantalla, mi archivo /res/xml/mis_preferencias.xml podría ser:



```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory
        android:title="@string/pref_sms_storage_title"
        android:key="pref_key_storage_settings">
        <CheckBoxPreference
            android:key="pref_key_auto_delete"
            android:summary="@string/pref_summary_auto_delete"
            android:title="@string/pref_title_auto_delete"
            android:defaultValue="false"... />
        <Preference
            android:key="pref_key_sms_delete_limit"
            android:dependency="pref_key_auto_delete"
            android:summary="@string/pref_summary_delete_limit"
            android:title="@string/pref_title_sms_delete"... />
        <Preference
            android:key="pref_key_mms_delete_limit"
            android:dependency="pref_key_auto_delete"
            android:summary="@string/pref_summary_delete_limit"
            android:title="@string/pref_title_mms_delete"... />
    </PreferenceCategory>
    ...
</PreferenceScreen>
```

Las etiquetas <....> que podéis usar en ese archivo son:

CheckBoxPreference

Representa un tipo de opción que sólo puede tomar dos valores distintos: activada o desactivada. Es el equivalente a un control de tipo *checkbox*. En este caso tan sólo tendremos que especificar los atributos: nombre interno de la opción (`android:key`), texto a mostrar (`android:title`) y descripción de la opción (`android:summary`). Veamos un ejemplo:

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

EditTextPreference

Representa un tipo de opción que puede contener como valor una cadena de texto. Al pulsar sobre una opción de este tipo se mostrará un cuadro de diálogo sencillo que solicitará al usuario el texto a almacenar. Para este tipo, además de los tres atributos comunes a todas las opciones (`key`, `title` y `summary`) también tendremos que indicar el texto a mostrar en el cuadro de diálogo, mediante el atributo `android:dialogTitle`. Un ejemplo sería el siguiente:

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

ListPreference

Representa un tipo de opción que puede tomar como valor un elemento, y sólo uno, seleccionado por el usuario entre una lista de valores predefinida. Al pulsar sobre una opción de este tipo se mostrará la lista de valores posibles y el usuario podrá seleccionar uno de ellos. Y en este caso seguimos añadiendo atributos. Además de los cuatro ya comentados (`key`, `title`, `summary` y `dialogTitle`) tendremos que añadir dos más, uno de ellos indicando la lista de valores a visualizar en la lista y el otro indicando los valores internos que utilizaremos para cada uno de los valores de la lista anterior (Ejemplo: al usuario podemos mostrar una lista con los valores "Español" y "Francés", pero internamente almacenarlos como "ESP" y "FRA").

Estas listas de valores se definen en `/res/values/arrays.xml` como dos arrays. Por ejemplo, el fichero `arrays.xml`:

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
    <string-array name="pais">
        <item>España</item>
        <item>Francia</item>
        <item>Alemania</item>
    </string-array>
    <string-array name="codigopais">
        <item>ESP</item>
        <item>FRA</item>
        <item>ALE</item>
    </string-array>
</resources>
```

En la preferencia ListPreference, utilizaremos los atributos `android:entries` y `android:entryValues` para hacer referencia a estas listas, como vemos en el ejemplo siguiente:

```
1 <ListPreference
2   android:key="opcion3"
3   android:title="Preferencia 3"
4   android:summary="Descripción de la preferencia 3"
5   android:dialogTitle="Indicar Pais"
6   android:entries="@array/pais"
7   android:entryValues="@array/codigopais" />
```

Aquí tenéis otro fichero ejemplo de preferencias:

```
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
  <PreferenceCategory android:title="Categoría 1">
    <CheckBoxPreference
      android:key="opcion1"
      android:title="Preferencia 1"
      android:summary="Descripción de la preferencia 1" />
    <EditTextPreference
      android:key="opcion2"
      android:title="Preferencia 2"
      android:summary="Descripción de la preferencia 2"
      android:dialogTitle="Introduce valor" />
  </PreferenceCategory>
  <PreferenceCategory android:title="Categoría 2">
    <ListPreference
      android:key="opcion3"
      android:title="Preferencia 3"
      android:summary="Descripción de la preferencia 3"
      android:dialogTitle="Indicar Pais"
      android:entries="@array/pais"
      android:entryValues="@array/codigopais" />
  </PreferenceCategory>
</PreferenceScreen>
```

La etiqueta `<PreferenceCategory>` es para agrupar en la pantalla las opciones que se muestren en grupos.

2. **Crear UN FRAGMENT que cargue este fichero XML y así se pueda mostrar esa pantalla al usuario.** Para crear un fragment hay dos opciones:
 - a) **En la carpeta de JAVA, BOTÓN DCHO->NEW JAVA CLASS y añadido en el código `extends PreferenceFragment` como se ve en el ejemplo de aquí abajo.**
 - b) **En la carpeta de JAVA, BOTÓN DCHO->NEW FRAGMENT->BLANK FRAGMENT y en el asistente que os aparece, DESMARCAR CREAR LAYOUT (no queremos layout).**

Tanto si escogéis opción a) como b), tendréis que copiar este código:

```
public static class SettingsFragment extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Load the preferences from an XML resource
        addPreferencesFromResource(R.xml.preferences);
    }
    ...
}
```

3. Crear UNA ACTIVITY que contenga el FRAGMENT que creasteis en el punto anterior.
Para ello el código será algo como:

```
public class SettingsActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Display the fragment as the main content.
        getFragmentManager().beginTransaction()
            .replace(android.R.id.content, new SettingsFragment())
            .commit();
    }
}
```

OJO: La activity que creamos aquí, NO LLEVA LAYOUT ASOCIADO, pues va a cargar el fragment.

Por eso lo normal, es crearla a mano, como una clase Java normal que extiende de Activity y sobrescribir el onCreate.

Si la creáis como siempre: BOTÓN DCHO-> NEW ACTIVITY, tenéis que asegurarnos que DESMARCAIS LA CASILLA DE CREAR EL LAYOUT, porque por defecto os crea uno y **FIJAROS QUE NO LLEVA setContentView()**.

Por tanto, **COMO CREAMOS A MANO LA ACTIVITY, EL REGISTRO EN EL MANIFEST, TAMBIÉN TIENE QUE SER A MANO**. Es decir, el paso

siguiente paso será añadir esta actividad al fichero *AndroidManifest.xml*, al igual que cualquier otra actividad que utilicemos en la aplicación.

```
1 <activity android:name=".SettingsActivity"
2     android:label="@string/app_name">
3 </activity>
```

4. **HACÉIS QUE un BOTÓN o UN MENÚ LANCE ESA ACTIVIDAD** cuando queráis que la pantalla de configuración se muestre al usuario (con un Intent como siempre).

CÓMO GRABAR LO QUE EL USUARIO ELIJA EN ESA PANTALLA ANTERIOR

NO HAY QUE HACER NADA.

SE GRABA AUTOMÁTICAMENTE EN UN FICHERO XML en la memoria interna de nuestro dispositivo.

Cada [Preference](#) que agregas tiene un par clave-valor correspondiente que el sistema utiliza para guardar la configuración en un archivo [SharedPreferences](#) predeterminado para las configuraciones de tu app. **Cuando el usuario cambia una configuración, el sistema actualiza el valor correspondiente en el archivo [SharedPreferences](#).** El único momento en el que debes interactuar directamente con el archivo [SharedPreferences](#) asociado es cuando necesitas **leer el valor para determinar el comportamiento** de tu app de acuerdo con la configuración del usuario.

El valor guardado en [SharedPreferences](#) para cada configuración puede ser uno de los siguientes tipos de datos:

- Booleano
- Flotante
- Int
- Largo
- String
- String [Set](#)

Ese fichero se guarda en esta ruta y con este nombre:

`/data/data/paquete.java/shared_prefs/paqueteJava_preferences.xml`

Si lo abris, con DOBLE CLICK, tendrá algo como:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
  <string name="nombre">prueba</string>
  <string name="email">modificado@email.com</string>
</map>
```

CÓMO LEER ESE FICHERO DE PREFERENCIAS (es decir, saber qué preferencias ha elegido el usuario)

Para obtener acceso a la colección de preferencias, necesitamos un objeto de la clase **SharedPreferences**.

Y se haría así; desde la activity donde queráis saber si el usuario ha elegido música o no, jugar en red o no, deberíais escribir algo como:

```
SharedPreferences sharedPref =
PreferenceManager.getDefaultSharedPreferences(this);
```

```
boolean miniaturasPref = sharedPref.getBoolean("miniaturas", true);
String texto = sharedPref.getString("nombre_usuario", "");
...
```

Como véis se leen con métodos “getString(), getBoolean(), getLong()”,... pasando como parámetros, la key de esa preferencia (lo que pusisteis en su xml como key) y un valor por defecto.

¿CÓMO USAR LAS PREFERENCIAS PARA GUARDAR OTROS DATOS QUE NO ESTÁN LIGADOS A UNA PANTALLA DE CONFIGURACIÓN,

como:

- Datos del usuario
- Puntuaciones
- Contadores que queremos almacenar por algún motivo,...
- ?

Para este uso, **NO NECESITAMOS CREAR NINGUNA PANTALLA DE CONFIGURACIÓN**, SÓLO NECESITAMOS ACCEDER AL FICHERO xml que guarda las preferencias y al que tenemos acceso a través de la clase `SharedPreferences` y conseguir escribir en él, desde nuestro código JAVA.

Por defecto, no se puede escribir en él directamente. **NECESITAMOS HACERLO MEDIANTE UN EDITOR QUE NOS OFRECE**. Así, el código sería algo como esto:

```
SharedPreferences pref=
    PreferenceManager.getDefaultSharedPreferences(this);
SharedPreferences.Editor editor=pref.edit();
editor.putBoolean("jubilado", true);
editor.putString("nombre", "Ana");
editor.put..... ("clave", valor);
editor.commit();
```

Esta forma de grabar en preferencias la necesitáis en vuestra tarea para grabar los datos del usuario al registrarlo, y la puntuaciones del juego.

BUENAS PRÁCTICAS EN EL USO DE PREFERENCIAS

Debido a que cuando se usan pantallas de configuración tenemos que consultar o leer muchas veces las preferencias a lo largo de nuestra app (en varias activities o fragments), es buena idea crear una clase java que incluya todos los métodos para gestionar las preferencias, así como los listeners que nos avisan de sus cambios (aunque para la tarea no os los exijo. Eso es más avanzado) y cualquier cosa que tenga que ver con ellas. Esa clase la implementamos con el patrón **SINGLETON** y así, cuando la queramos usar, haremos algo como:

MiClase.getInstance().miMetodo().

Un ejemplo de clase SINGLETON para este uso podría ser:

```
public class Preferencias {

    private static final Preferencias INSTANCE = new Preferencias();
    private SharedPreferences pref;
    private Context context;

    public static Preferencias getInstance() {
        return INSTANCE;
    }

    private Preferencias() {

        /**
         * Method gets an instance of SharedPreferences and set up the context
         *
         * @param context the context
         */
        public void inicializa(Context context) {
            pref = PreferenceManager.getDefaultSharedPreferences(context);
            this.context = context;
        }

        /**
         * Update summary when SharedPreferences changes
         *
         * @param sharedPreferences the user sharedPreferences
         * @param pref the preference which has changed
         */
        public void updatePrefsSummary(SharedPreferences sharedPreferences,
                                       Preference pref) {

            String originalSummary = "";

            if (pref == null)
                return;
        }
    }
}
```



```
//get the original summary of one preference (as it appears in xml
file)
    int indexOfBracket = pref.getSummary().toString().indexOf('(');
//if it has changed, it will have a bracket
    if (indexOfBracket != -1)
        originalSummary = pref.getSummary().toString().substring(0,
indexOfBracket);
    else
        originalSummary = pref.getSummary().toString();

    if (pref instanceof ListPreference) {
        // List Preference
        ListPreference listPreference = (ListPreference) pref;
        listPreference.setSummary(originalSummary + " (" +
listPreference.getEntry() + ")");

    } else if (pref instanceof EditTextPreference) {
        // EditTextPreference
        final EditTextPreference editTextPreference =
(EditTextPreference) pref;
        editTextPreference.setSummary(originalSummary + " (" +
editTextPreference.getText() + ")");

        if (isAsteroidsFragmentsPreference(editTextPreference)) {
            //It's fragments preference of my game
            checkFragmentsPreferenceCorrectValues(editTextPreference,
originalSummary);
        } else if (isMaxNumberPlayersPreference(editTextPreference)) {
            //It's maximum number of players preference
            checkMaxNumberPlayersPrefenceCorrectValues(editTextPreference);
        }

    }

}

/*
 * Init summary
 * Method initializes preference summary for all preferences when
activity starts
 */
public void initSummary(PreferenceScreen preferenceScreen) {
    int numberPreferences = preferenceScreen.getPreferenceCount();
    for (int i = 0; i < numberPreferences; i++) {
        initPreferencesSummary(pref,
preferenceScreen.getPreference(i));
    }
}

/*
 * Init single Preference
 */
private void initPreferencesSummary(SharedPreferences
sharedPreferences,
                                   Preference p) {
    if (p instanceof PreferenceCategory) {
```

```
        PreferenceCategory preferenceCategory = (PreferenceCategory) p;
        int numberPreferences =
preferenceCategory.getPreferenceCount();
        for (int i = 0; i < numberPreferences; i++) {//for each
preference in this category group
            initPreferencesSummary(sharedPreferences,
preferenceCategory.getPreference(i));
        }
    } else {
        updatePrefsSummary(sharedPreferences, p);
    }
}

/**
 * Method checks if this preference is asteroids fragments preference
 *
 * @param editTextPreference EditTextPreference to check
 * @return true if editTextPreference is an asteroids fragments
preference
 */
public boolean isAsteroidsFragmentsPreference(EditTextPreference
editTextPreference) {
    if
(editTextPreference.getKey().equals(context.getString(R.string.pref_fragmen
tos_key))) {
        return true;
    }
    return false;
}

/**
 * Method checks if this preference is maximum number of players
preference
 *
 * @param editTextPreference EditTextPreference to check
 * @return true if editTextPreference is a maximum number of players
preference
 */
public boolean isMaxNumberPlayersPreference(EditTextPreference
editTextPreference) {
    if
(editTextPreference.getKey().equals(context.getString(R.string.pref_max_num
_jugadores_key))) {
        return true;
    }
    return false;
}

/**
 * Method checks if values introduced in this preference are correct.
In this case, they will be saved.
 * In other case, a message will be showed to inform about that
situation
 *
 * @param editTextPreference preference to check
 * @param initialSummary summary which appears in xml file
initially
```

```
*/
public void checkFragmentsPreferenceCorrectValues(final
EditTextPreference editTextPreference, final String initialSummary) {

    editTextPreference.setOnPreferenceChangeListener(new
Preference.OnPreferenceChangeListener() {
    @Override
    public boolean onPreferenceChange(Preference preference, Object
newValue) {
        int value;
        try {
            value = Integer.parseInt((String) newValue);
        } catch (Exception e) {
            Toast.makeText(context,
context.getString(R.string.fragmentPreferenceNotNumber),
Toast.LENGTH_SHORT).show();
            return false;
        }
        if (value >= 0 && value <= 9) {

            editTextPreference.setSummary(initialSummary + " (" +
value + ")");
            return true;
        } else {
            Toast.makeText(context,
context.getString(R.string.fragmentPreferenceIncorrect),
Toast.LENGTH_SHORT).show();
            return false;
        }
    }
});
}

/**
 * Method checks if values introduced in this preference are correct.
 * In this case, they will be saved.
 * In other case, a message will be showed to inform about that
 * situation
 */
@param editTextPreference preference to check
*/
public void checkMaxNumberPlayersPreferenceCorrectValues(final
EditTextPreference editTextPreference) {
    editTextPreference.setOnPreferenceChangeListener(new
Preference.OnPreferenceChangeListener() {
    @Override
    public boolean onPreferenceChange(Preference preference, Object
o) {
        int valor;
        try {
            valor = Integer.parseInt((String) o);
        } catch (Exception e) {
            Toast.makeText(context,
context.getString(R.string.maximoNumeroJugadoresNotNumber),
Toast.LENGTH_SHORT).show();
            return false;
        }
        if (valor >= 1 && valor <= 5) {

            editTextPreference.setSummary(context.getString(R.string.maximoNumeroJugado
```

```
resUpdatedSummary) + "(" + valor + ")";
        return true;
    } else {
        Toast.makeText(context,
context.getString(R.string.maximoNumeroJugadoresIncorrect),
Toast.LENGTH_SHORT).show();
        return false;
    }
    }
});
}
```

```
public int criterioSeleccion() {
    return parseInt(pref.getString("seleccion", "0"));
}

public String tipoSeleccion() {
    return (pref.getString("tipo seleccion", "BAR"));
}

public String criterioOrdenacion() {
    return (pref.getString("orden", "valoracion"));
}

public int maximoMostrar() {
    return parseInt(pref.getString("maximo", "50"));
}

}

}
```