Shared Preferences (1)

La clase *SharedPreferences* proporciona los métodos necesarios para almacenar y recuperar fácilmente los datos como pares clave-valor, con la limitación de que sólo permite el uso de datos primitivos: *int*, *float*, *String*, *Parcelable*, etc.

Los datos se mantendrán hasta que la aplicación sea desinstalada.

3

Shared Preferences (2)

Para utilizar *SharedPreferences* deberemos recuperar una instancia de la clase con uno de los métodos:

- *getPreferences (int mode)*: no podemos especificar el nombre del archivo donde se almacenarán los datos, así que utilizaremos este método cuando necesitemos un único archivo de preferencias por Actividad.
- *getSharedPreferences (String name, int mode)*: podemos especificar el nombre del archivo, así que utilizaremos este método cuando necesitemos varios archivos, o queramos compartir los valores entre diferentes Actividades.

4

Shared Preferences (3)

El parámetro *mode* puede tomar los valores:

- *MODE_PRIVATE* (defecto): el archivo es privado y es el modo de funcionamiento por defecto.
- *MODE_APPEND*: permite añadir las nuevas preferencias a las ya existentes.
- *MODE_MULTI_PROCESS*: comprobará si se han producido modificaciones en las preferencias después de haber sido cargadas.
- *MODE_WORLD_READABLE/MODE_WORLD_WRITEABLE*: el resto de aplicaciones puede acceder al archivo para leer o escribir. Por razones de seguridad se han declarado obsoletos.
- *MODE_ENABLE_WRITE_AHEAD_LOGGING*: habilita el modo de escritura anticipada.

5

Shared Preferences (4)

- Leer datos

Una vez obtenida la instancia de *SharedPreferences*, podemos utilizar sus métodos *get* para obtener los datos almacenados a partir de su clave: *getBoolean*, *getInt*, *getFloat*, *getString*, etc.

En estos métodos se puede establecer como segundo parámetro un valor por defecto que se utilizará si el dato no existe en el archivo de *SharedPreferences* consultado.

```
SharedPreferences misDatos = getPreferences(MODE_PRIVATE);
Boolean hayDatos = misDatos.getBoolean("hayDatos", false);
```

6

Shared Preferences (5)

- Escribir datos

Para añadir datos a un *SharedPreferences (SP)*, tenemos que obtener una instancia de la clase *Editor* llamando al método *edit()* desde nuestro objeto *SP*.

A continuación, utilizaremos el método put adecuado para agregar los datos: *putBoolean*, *putInt*, *putFloat*, *putString*, etc.

Una vez que hayamos agregado todos los datos, utilizaremos el método *apply()* sobre el editor para hacer efectiva la escritura de los datos.

```
SharedPreferences.Editor editor = misDatos.edit();
editor.putBoolean("hayDatos", true);
editor.apply();
```

7

Shared Preferences (6)

- Eliminar datos

Podemos eliminar un valor de nuestra colección *SharedPreferences* utilizando el método *remove()* sobre el editor.

Igual que ocurre al agregar datos, también tendremos que llamar al método *apply()* para que los cambios tengan efecto.

```
SharedPreferences.Editor editor = misDatos.edit();
editor.remove("hayDatos");
editor.apply();
```

8

Shared Preferences (7)

- Suscribirse a los cambios

También podemos hacer que una aplicación se suscriba a los cambios realizados en la colección:

1) Declaramos el *listener OnSharedPreferenceChangeListener* como atributo de nuestra clase.
2) Creamos el *listener* e implementamos su método *onSharedPreferenceChanged*.
3) Registramos el *listener* y se lo asignamos a nuestro objeto *SharedPreferences* mediante el método:
   registerOnSharedPreferenceChangeListener
4) Si en algún momento deseamos anular la suscripción utilizamos el método:
   unregisterOnSharedPreferenceChangeListener

9

Shared Preferences (8)

- Suscribirse a los cambios

```
public class MainActivity extends AppCompatActivity {
  protected SharedPreferences misDatos;
  protected SharedPreferences.OnSharedPreferenceChangeListener prefListener;
  ...
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ...
    misDatos = getPreferences(MODE_PRIVATE);
    prefListener = new SharedPreferences.OnSharedPreferenceChangeListener() {
      @Override
      public void onSharedPreferenceChanged(SharedPreferences sp, String s) {
        ...
      }
    };
    misDatos.registerOnSharedPreferenceChangeListener(prefListener);
    ...
  }
}
```

10

Shared Preferences (9)

- Aplicaciones

Mediante *SharedPreferences* podemos almacenar cualquier dato simple de utilidad para nuestra app.

Sin embargo, como su propio nombre indica uno de los usos más habituales es almacenar las preferencias de la aplicación.

Para facilitar la gestión de las preferencias, tenemos las clases *PreferenceActivity* y *PreferenceFragment*, que nos permiten crear pantallas de preferencias desde las que editar y almacenar fácilmente estos valores haciendo uso de *SharedPreferences*:

https://developer.android.com/reference/android/preference/PreferenceActivity.html
https://developer.android.com/reference/android/preference/PreferenceFragment.html

11

Shared Preferences (10)

- Aplicaciones

A partir de la API 29 (Android 10) la gestión de las preferencias se realiza mediante la biblioteca *Preference* de AndroidX:

https://developer.android.com/guide/topics/ui/settings

12