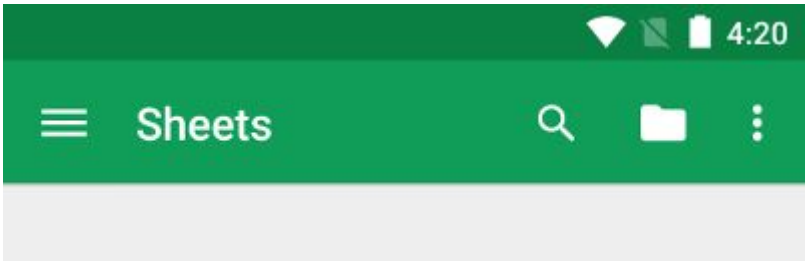
ActionBar

Seminario de lenguajes Android

Introducción

El objetivo de este documento es brindar los conceptos básicos para la definición de acciones en la ActionBar de nuestras aplicaciones Android.

La ActionBar, también conocida como barra de aplicación, es un elemento de diseño ubicado en la parte superior de nuestra aplicación. El principal objetivo de esta barra es proveer una manera estandarizada dentro del ecosistema Android para mostrar aspectos de navegabilidad, la identidad de nuestra aplicación y brindar acceso a acciones importantes dentro de la misma.



Configuración

Para definir las diferentes acciones disponibles en la *ActionBar* deberemos desarrollar en primer lugar un recurso de menú donde se definan las opciones disponibles:

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

   <item
       android:id="@+id/menu_item_1"
       android:title="Item 1"
       android:icon="@drawable/ic_search"
       app:showAsAction="ifRoom" />

   <item
       android:id="@+id/menu_item_2"
       android:title="Item 2"
       android:icon="@drawable/ic_search"
       app:showAsAction="never" />

</menu>
```

Notar que cada opción del menú contiene un atributo *showAsAction*. Este atributo permite definir si la opción estará disponible como un ícono en la barra (*ifRoom*) o directamente irá a parar al menú desplegable (*never*).

Los íconos de la barra serán visibles solo si hay espacio disponible, en caso contrario, se mostrarán también en el menú desplegable.

Una vez definido nuestro recurso de menú, es necesario hacer referencia al mismo redefiniendo el método *onCreateOptionsMenu* de la siguiente manera:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.mi_recurso_de_menu, menu);

    return super.onCreateOptionsMenu(menu);
}
```

En el código se puede apreciar que se está leyendo el recurso definido en el paso anterior: `R.menu.mi_recurso_de_menu`.

Con estos dos pasos, nuestra ActionBar ya tiene definidas nuestras acciones. Lo único que nos queda es darles comportamiento redefiniendo el método *onOptionsItemSelected* en nuestra Activity:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_1:
            // El usuario hizo click en la primer action
            return true;

        case R.id.menu_item_2:
            // El usuario hizo click en la segunda action
            // as a favorite...
            return true;

        default:
            // No se pudo reconocer la acción elegida por el usuario.
            // Invocar a la superclase para que se encargue.
            return super.onOptionsItemSelected(item);
    }
}
```

Acciones predefinidas

Existen ciertas acciones comunes dentro del ecosistema Android que pueden ser necesarias en nuestras aplicaciones. Si bien es posible implementar estas acciones desde cero, a veces es conveniente utilizar las implementaciones propuestas por la plataforma.

Android provee el concepto de *ActionProvider* como mecanismo de definición y reutilización de acciones para nuestras *ActionBar*.

A modo de ejemplo se mostrará cómo utilizar *ShareActionProvider* para mostrar una acción en la *ActionBar* de nuestra aplicación que nos permita mostrar un botón para compartir contenido con otras aplicaciones.

En primer lugar, en nuestros recursos de menú, deberemos establecer el ítem de menú de la siguiente manera:

```
<item android:id="@+id/action_share"
    android:title="Compartir"
    app:showAsAction="ifRoom"
app:actionProviderClass="android.support.v7.widget.ShareActionProvider"/>
```

El atributo *actionProviderClass* contiene el nombre de la clase que implementará el comportamiento. En este caso estamos utilizando la implementación de *ShareActionProvider* de la librería de compatibilidad de Android.

Notar que no es necesario especificar un ícono para el ítem, ya que el mismo *ShareActionProvider* se encargará de definirlo.

Una vez definido nuestro recurso de menú, podemos configurar su comportamiento en el método *onCreateOptionsMenu* de la siguiente manera:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.mi_recurso_de_menu, menu);

    // Obtener el item de menu que se desea configurar
    MenuItem item = menu.findItem(R.id.action_share);

    // Obtener el ShareActionProvider asociado al item de menu
    ShareActionProvider shareActionProvider = (ShareActionProvider)
MenuItemCompat.getActionProvider(item);

    // Guardar instancia de ShareActionProvider en una variable de
instancia de la Activity
    this.shareActionProvider = shareActionProvider;

}
```

En el código se puede observar que se está obteniendo la instancia de *ShareActionProvider* mediante la llamada a *MenuItemCompat.getActionProvider(item)*.

Una vez obtenida la instancia, es posible configurarla en el momento o incluso guardarla en una variable de instancia para configurarla más adelante.

A modo de ejemplo se muestra como configurar el *Intent* que se ejecutará cuando el usuario desee realizar la acción de compartir:

```
public void compartir() {
    Intent i = new Intent();
    i.setAction(Intent.ACTION_SEND);
    i.putExtra(Intent.EXTRA_TEXT, “Texto a compartir”);

    this.shareActionProvider.setShareIntent(i);
}
```

Notar que en el caso de los *ActionProvider* no es necesario redefinir el método *onOptionsItemSelected* dado que el *ActionProvider* ya llevó a cabo ese comportamiento por nosotros. En el caso particular del *ShareActionProvider* solo hace falta configurar el *Intent* que queremos ejecutar al compartir.

Referencias

- https://developer.android.com/training/appbar
- https://developer.android.com/reference/android/support/v7/widget/ShareActionProvider