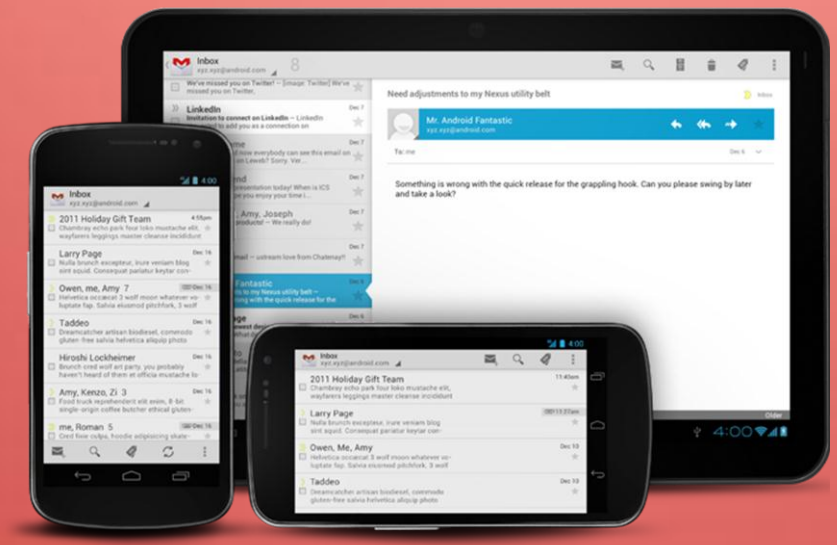


ACTIONBAR COMPACT

Héctor Abad Omaña

2ºDAM-Curso 2013/2014

3/02/2014

ÍNDICE

1. Introducción a ActionBar

1.1. ¿Qué es ActionBar?

La ActionBar de Android es la barra de título y herramientas que aparece en la parte superior de muchas aplicaciones actuales. Normalmente muestra un icono, el título de la actividad en la que nos encontramos, una serie de botones de acción, y un menú desplegable donde se incluyen más acciones que no tienen espacio para mostrarse como botón o simplemente no se quieren mostrar como tal.

La barra de acción proporciona varias funciones clave:

- Proporciona un espacio dedicado para darle a la aplicación una identidad y que indica la ubicación del usuario en la aplicación.
- Hace acciones importantes y accesibles de una manera intuitiva.
- Soporta navegación consistente y vista dentro de aplicaciones (con fichas o listas desplegables).

1.2. Ejemplo visual

El siguiente ejemplo nos muestra la barra de acción de WhatsApp:

1.3. Uso de la ActionBar en dispositivos anteriores a Android 3.0 T

La ActionBar apareció con el Android 3.0 (API11), por ello, para configurar la ActionBar en móviles que soportan una versión anterior hay dos formas:

- Android soporta una librería que se denomina **ActionBarSherlock** que es compatible con versiones Android de la 2.0 en adelante y que se descarga de la web http://actionbarsherlock.com/ .Hay que decir que esta poco a poco perdiendo importancia y que se utiliza el siguiente punto.
- También existe **ActionBarCompat**, que se explicara más adelante, ya que es la forma que se está utilizando actualmente.

2. Ubicación y Atributos

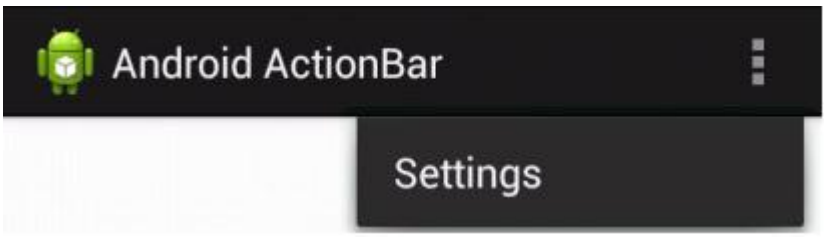
2.1. Ubicación de ActionBar

La ubicación de los menús se encuentra en la carpeta ***res/menu*** mediante un fichero XML. El menú se compone de un elemento raíz ***<menu>*** y dentro de ese menú definiremos los ***<items>*** que serán las opciones de ese menú. La salida por defecto es la siguiente:

main.xml

```
1  <menu xmlns:android="http://schemas.android.com/apk/res/android" >
2
3      <item
4          android:id="@+id/action_settings"
5          android:orderInCategory="100"
6          android:showAsAction="never"
7          android:title="@string/action_settings"/>
8
9  </menu>
10
```

Al crear un proyecto con las últimas versiones de ADT en Eclipse, se genera automáticamente un menú, siempre que ejecutemos con versiones superiores a la 3.0. En este menú solo aparecen los tres puntitos en la parte superior derecha y dentro de él ***"Settings".*** Esta configuración viene por defecto.



¿Por qué no nos aparece el menú overflow de la parte superior?

Android establece que si el dispositivo tiene un botón físico de menú, el botón de overflow desaparece y se sustituye por un menú tradicional que aparece en la zona inferior cuando se pulsa el botón menú del dispositivo. Este será el caso de la mayoría de dispositivos que funcionen con Android 2.x. Si pulsamos el botón menú del emulador veremos como ahora sí aparece nuestra acción ***"Settings".***

2.2. Atributos de ActionBar

Los atributos más importantes de la ActionBar son:

- ***android:id*** :El ID identificativo del elemento, con el que podremos hacer referencia dicha opción.
- ***android:title*** : El texto que se visualizará para la opción.
- ***android:icon*** : El icono asociado a la acción.
- ***android:showAsAction*** : Si se está mostrando una ActionBar, este atributo indica si la opción de menú se mostrará como botón de acción o como parte del menú de overflow.

El ***showAsAction*** admite diferentes valores, son los siguientes:

ifRoom	Muestra el icono si hay espacio disponible en la pantalla
never	La opción siempre se mostrará como parte del menú de overflow.
always	La opción siempre se mostrará como botón de acción. Este valor puede provocar que los elementos se solapen si no hay espacio suficiente para ellos.
WithText	Se mostrará el texto de la opción junto al icono en el caso de que éste se esté mostrando como botón de acción.
collapseActionView	Define la disposición de acción asociado. Este punto de vista de acción define utilizando **android: actionLayout** o **android: actionViewClass**

3. Uso de ActionBar

3.1. Creación de la barra de acción

Como hemos visto anteriormente, por defecto, nuestra aplicación ya contiene un menú, pero ¿cómo podemos crear nuestra barra de acción?

Pasos:

- Creamos un nuevo ***<ítem>*** como el que viene por defecto.
- El ***android:id*** es muy importante ya que el nombre que le demos será el que utilizamos al llamarlo en nuestra activity principal.
- IMPORTANTE: El ***android:title*** es un string que tiene que estar creado en ***values/strings*** sino nos dará un error.
- El ***android:icon*** es la imagen. Podemos utilizar las que vienen definidas en Android o una imagen personaliza
    - Si queremos utilizar una que viene en Android la instrucción es :

```
android:icon="@android:drawable/ic_ y el logo que deseemos "
```

    - Pero si queremos utilizar una personal , lo primero será tenerla en la carpeta ***res/drawable-mdpi*** ya que estas imágenes no son muy grandes y la instrucción es:

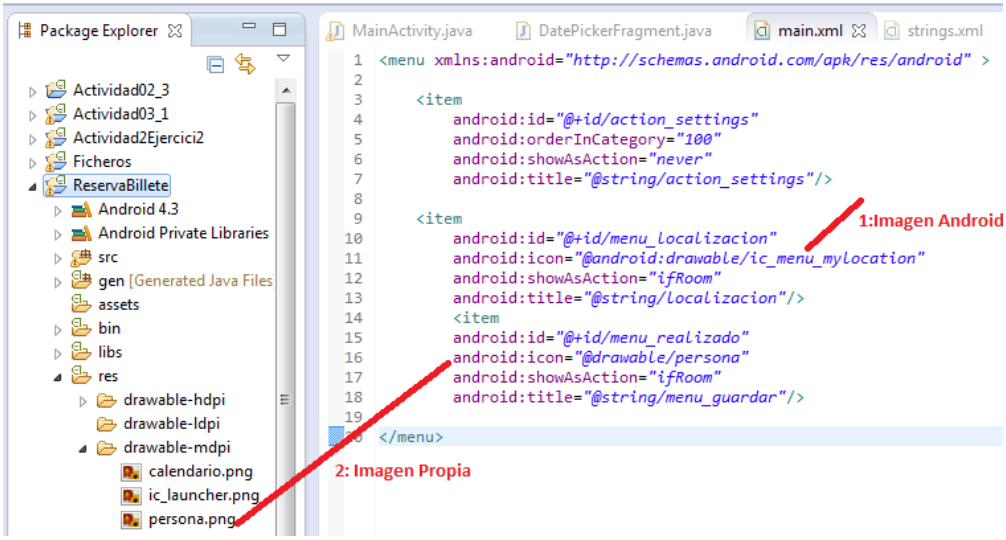
```
android:icon="@drawable/ el nombre de la imagen"
```

- Y por último el ***showAsAction*** que ya está explicado arriba.

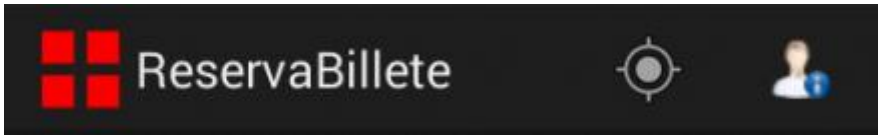
3.2. Ejemplo visual

En el ejemplo voy a utilizar una imagen del sistema Android y otra mía propia.

La primera será un icono de localización y la segunda un icono de persona.



La salida seria la siguiente:



3.3. Asociación y pulsación en la actividad principal

Una vez que tenemos definido nuestro XML lo que debemos de hacer es asociarlo en nuestra main principal para que tenga utilidad.

Asociamos nuestro XML a la actividad principal, esto se hace en el método ***onCreateOptionsMenu()*** en el cual debemos inflar el menú con la instrucción ***inflate().*** Esta parte ya viene definida al crear un nuevo proyecto con Eclipse.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
```

Donde ***R.menu.main*** es el nombre que tiene el fichero XML definido en ***res/menu***

¿Y cómo hacemos para que nuestros botones tengan utilidad cuando se pulsen?

Lo que debemos de hacer es sobrescribir el método ***onOptionsItemSelected().*** Aquí es donde tenemos que utilizar el ***"android:id"*** que hemos asignado a cada ítem del menú en el XML. La forma de saber que botón se ha pulsado lo sabemos con el método ***getItemId().*** En mi ejemplo seria así:

```java
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_realizado:
            Log.i("ActionBar", "Realizado");;
            Toast toast1 = Toast.makeText(getApplicationContext(),
                    "Realizado por: Héctor Abad Omaña", Toast.LENGTH_SHORT);
            toast1.show();

            return true;
        case R.id.menu_localizacion:
            Log.i("ActionBar", "Localizacion");;
            Toast toast2 = Toast.makeText(getApplicationContext(),
                    "Se encuentra en : IES SAN ANDRES", Toast.LENGTH_SHORT);
            toast2.show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

En este caso he utilizado un Toast para saber que me funciona correctamente.

4. ActionBar Compact

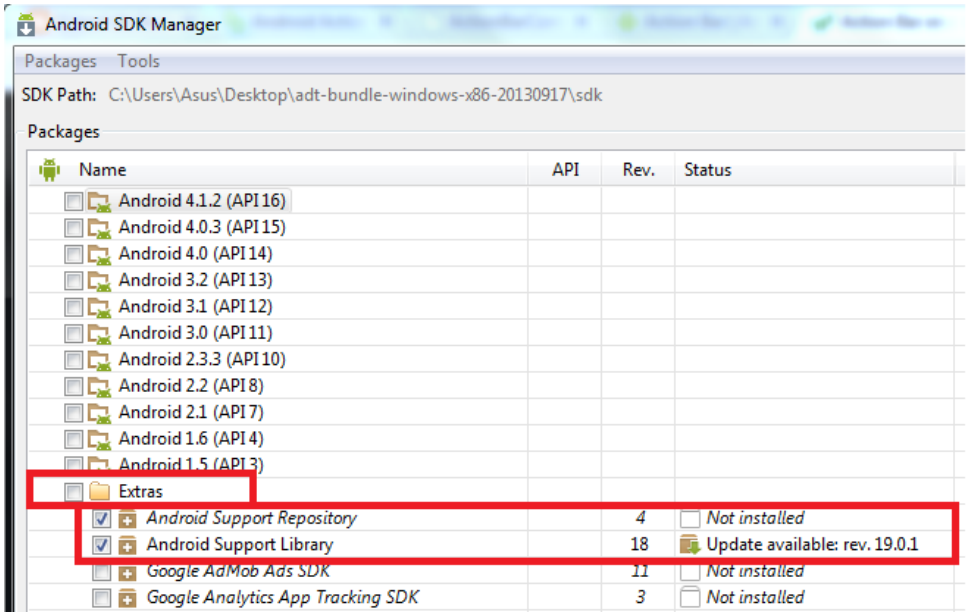
4.1. ¿Qué es ActionBar Compact?

Anteriormente hemos visto que la ActionBar solo la podíamos implementar en versiones posteriores a la 3.0 de Android, y para abarcar las versiones anteriores teníamos que hacer uso de librerías no oficiales de Google como lo es la librería de **ActionBar Sherlock**, pero Google ha liberado esta librería de compatibilidad para utilizar en versiones anteriores a la 3.0 de Android la ActionBar.

4.2. Pasos de configuración de las librerías apropiadas

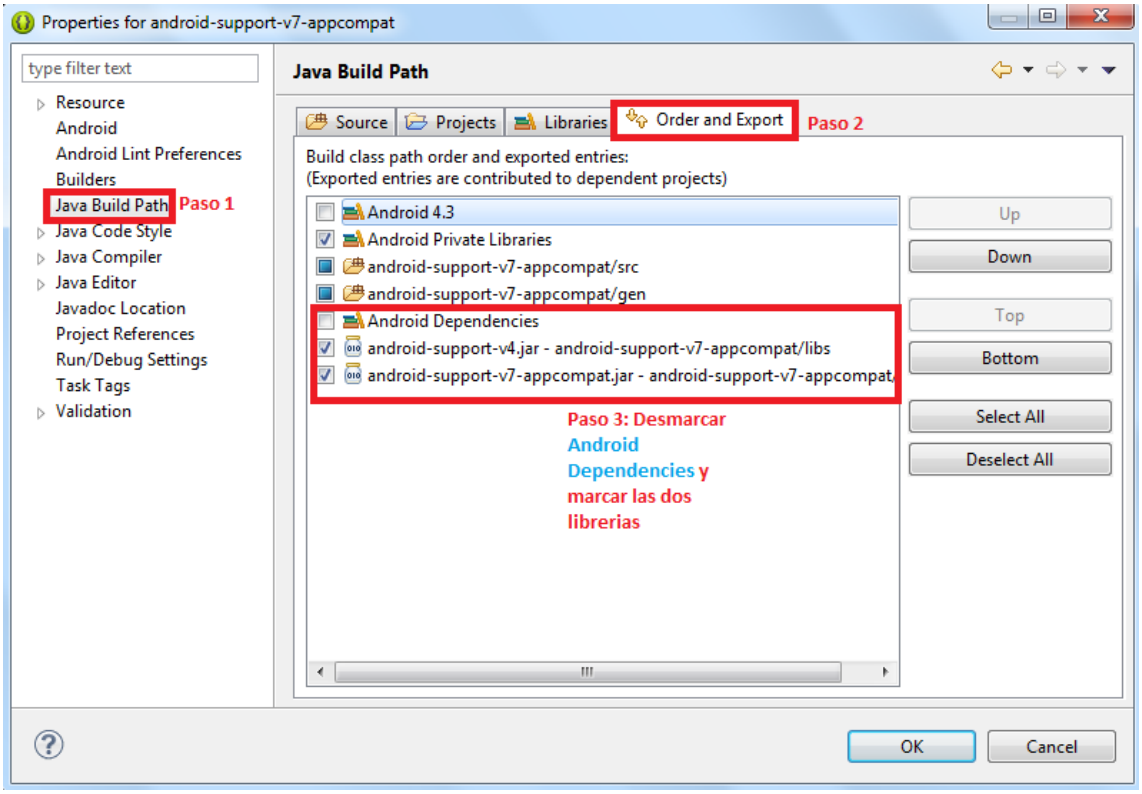
Para poder utilizar esta librería tenemos que realizar los siguientes pasos:

1. Iniciamos el Android ***SDK Manager.***
2. Nos colocamos en la pestaña ***Extras*** (esta al final) y seleccionamos ***Android Support Library***.
3. Instalamos los paquetes.



4. Una vez que hemos actualizado los repositorios de Android, entramos en Eclipse y seleccionamos ***File/Import***.
5. Seleccionamos ***Android/Existing Android Code Into WorkSpace.***
6. Para este paso tenemos que tener la carpeta donde se nos descargó el Eclipse porque es aquí donde están las bibliotecas de compatibilidad. Una vez que sepamos donde esta seleccionamos ***Browse*** y la ruta es la siguiente ***<sdk> / extras/android/support/v7/appcompat /*** .En esta pantalla también debemos seleccionar ***Copy projects into Workspace*** y finalizamos.
7. Se nos ha creado un nuevo proyecto. Ahora lo que tenemos que hacer es expandir la carpeta ***libs*** y sobre las dos librerías haremos click derecho y seleccionamos ***Build Path > Add to BuildPath***.

8. Ahora haremos click derecho en la carpeta del proyecto y seleccionamos ***Build Path > Configure Build Path.***
9. En la pestaña ***Order and Export*** seleccionamos las dos libreras ***.jar*** y deseleccionamos ***Android Dependences.***



10. Para finalizar pulsamos ***OK.***

4.3. Como añadir las librerías a un proyecto

Ya tenemos un proyecto creado para la utilización de nuestra librería de compatibilidad en cualquier proyecto.

Para añadir estas librerías a un proyecto lo que tenemos que hacer es:

1. Click derecho en el nuevo proyecto y pulsamos ***Propierties***.
2. Nos fijamos en la parte izquierda y seleccionamos ***Android.***
3. Ahora, en el panel de la librería, pulsamos ***Add*** y por defecto nos aparece el proyecto que hemos creado anteriormente.
4. Para finalizar ***Apply*** y ***Finish***

4.4. Tres pasos para el correcto funcionamiento del proyecto

Para que nuestro proyecto pueda estar totalmente configurado nos quedan tres pasos:

- Tendremos que sustituir (o extender) el tema visual utilizado en la aplicación por alguno de los definidos específicamente para la librería de compatibilidad(esto lo tenemos que hacer en el AndroidManifest) :

- Theme.AppCompat
- Theme.AppCompat.Light
- Theme.AppCompat.Light.DarkActionBar

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/Theme.AppCompat.Light.DarkActionBar" >
```

- Tenemos que modificar también la clase de la que heredan nuestras actividades, sustituyendo la clase Activity por la nueva clase ***ActionBarActivity.***
- En versiones anteriores a Android 3.0 no existían los atributos de los menús destinados a definir la funcionalidad de la ActionBar, por ejemplo el atributo ***android:showAsAction***, no podemos utilizarlos libremente ahora que queremos que nuestra aplicación funcione sobre dichas versiones. Para solucionar esto lo que haremos será utilizar dichos atributos indicando un espacio de nombres personalizado, que definiremos en el elemento ***<menu>*** y posteriormente usaremos en los elementos ***<item>.*** Pasos:
  - Creamos un nuevo espacio de nombres:

xmlns:***nombre_del_nuevo_espacio_de_nombres***=”http://schemas.android.com/apk/res-auto”

Donde el nombre del nuevo espacio puede ser cualquier nombre.

Por ejemplo:

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:hectorabad="http://schemas.android.com/apk/res-auto" >
```

5. Personalización y Retroceso de nuestra ActionBar

5.1. Personalización ActionBar

La ActionBar tiene unos métodos que son muy útiles, y para poder utilizarlos tenemos que instanciar al objeto ActionBar:

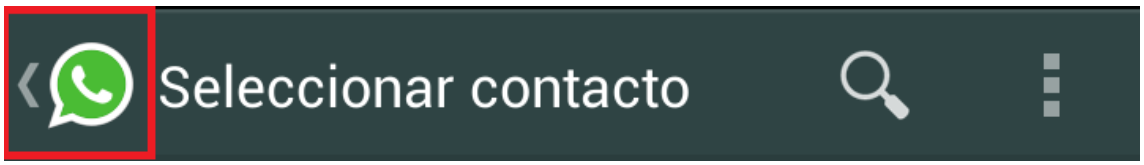
```
ActionBar actionBar = getSupportActionBar();
```

Estos son los métodos más útiles:

```
actionBar.hide();                          //Ocultar ActionBar

actionBar.setIcon(Drawable);        //Establecer icono

actionBar.setTitle(CharSequence);      //Establecer título el titulo

actionBar.setSubtitle(CharSequence);     //Establecer Subtitulo
```

5.2. Como retroceder de una actividad a otra mediante la ActionBar

La ActionBar también ofrece la posibilidad de la navegación de vuelta cuando la aplicación implica una relación jerárquica entre las pantallas. Un pequeño icono de la flecha hacia atrás se mostrará:



Para activar esta opción tenemos que seguir estos dos pasos:

- Activar el botón volver en la ActionBar a través del siguiente método. Esto se escribirá en el método ***onCreate*** de la actividad secundaria :

```
actionBar.setDisplayHomeAsUpEnabled(true);
```

- Indicar a que actividad volverá después de pulsar el botón volver, para ello tenemos dos posibilidades de hacerlo:
    - Especificar la actividad padre en el AndroidManifest(es la mejor opción cuando la actividad padre siempre es la misma, a continuación se muestra un ejemplo) . Para apoyar a los dispositivos más antiguos con la biblioteca de compatibilidad, también incluirá una ***<meta-data>*** elemento que especifica la actividad de los padres como el valor para ***android.support.PARENT_ACTIVITY***

```
<activity
    android:name="com.datohosting.ejemploactionbar.CrearTabsSwipe"
    android:parentActivityName="com.datohosting.ejemploactionbar.MenuPrincipal">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.datohosting.ejemplolistview.MenuPrincipal" />
</activity>
```

- Reemplazar los métodos ***getSupportParentActivityIntent()*** y ***onCreateSupportNavigateUpTaskStack()*** en nuestra activity (esta forma es apropiada cuando la actividad padre puede ser diferente).

6.Bibliografia

Páginas relacionadas con ActionBar:

http://www.sgoliver.net/blog/?p=3620

http://developer.android.com/guide/topics/ui/actionbar.html

http://www.androidhive.info/2013/11/android-working-with-action-bar/

http://www.vogella.com/tutorials/AndroidActionBar/article.html

http://www.elandroidelibre.com/2013/11/desarrollando-android-6-action-bar-y-listeners.html

Páginas relacionadas con ActionBar Compact:

http://androideity.com/2013/10/09/android-actionbar-compact/

http://antonioleiva.com/actionbarcompat-how-to-use/

http://developer.android.com/tools/support-library/setup.html#libs-with-res

http://www.sgoliver.net/blog/?p=4088

Explicación de todos los tipos de menús y código detallado:

http://elbauldeandroid.blogspot.com.es/2013/10/actionbar-android-en-contruccion.html