Toasts

Una "toast notification" es un mensaje que aparece en forma de "pop-up" mientras que la Activity que esta en pantalla en ese momento se mantiene visible y activa.

El mensaje desaparece automáticamente después de un tiempo prefijado y no acepta interacción con el usuario.

Ejemplo:

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_LONG;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

El método estático "makeText()" recibe 3 parámetros, el contexto, un texto del tipo CharSequence , y la duración del mensaje en la pantalla, para esta duración, existen dos constantes predefinidas:

- LENGTH_SHORT
- LENGTH_LONG

Posición en la pantalla

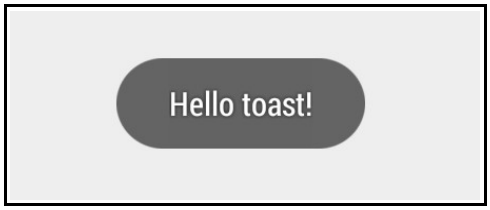
La posición por defecto del diálogo es cerca de la parte de abajo de la pantalla, centrado en forma horizontal. Esta posición se puede cambiar mediante el método "setGravity()". Este método posee tres parámetros, el primero, recibe constantes definidas en la clase Gravity:

- TOP
- BOTTOM
- LEFT
- RIGTH
- CENTER

El segundo y tercer parámetro, indican un offset en x e y respectivamente, de la posición prefijada por el primer parámetro.

Ejemplo:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```



Creando un Toast Personalizado

El objeto Toast, posee un método llamado "setView()", el cual recibe un objeto de tipo View el cual será el objeto mostrado en pantalla. Por esta razón, solo basta con crear un archivo xml de layout, el cual contendrá la estructura que se desea mostrar en el Toast.

Ejemplo: (creando un layout "toast_layout_root")

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="horizontal"
              android:padding="16dp"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:background="#FF0000">


    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:src="@drawable/ic_launcher" />

    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="fill_parent"
              android:textColor="#FFF"/>

</LinearLayout>
```

En este ejemplo creamos un layout con fondo rojo, y que contiene un objeto ImageView, y un objeto TextView.

El código java para disparar este Toast es :

```
        LayoutInflater inflater = getLayoutInflater();
        View layout = inflater.inflate(R.layout.toast_layout_root,null);

        TextView text = (TextView) layout.findViewById(R.id.text);
        text.setText("Mensaje Toast");

        Toast toast = new Toast(getApplicationContext());
        toast.setDuration(Toast.LENGTH_LONG);
        toast.setView(layout);
        toast.show();
```

Primero, creamos el objeto View a partir del archivo xml, para ello utilizamos el objeto "inflater" que nos proporciona la Activity:

```
        LayoutInflater inflater = getLayoutInflater();
```

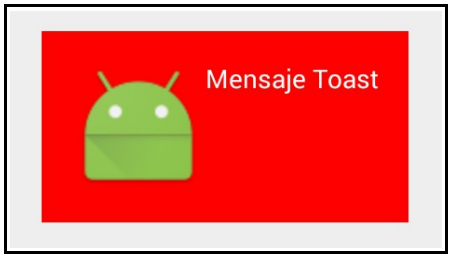
A él le pasamos el id del archivo de layout que definimos con anterioridad, en el metodo "inflate()" de modo de obtener el objeto View:

```
        View layout = inflater.inflate(R.layout.toast_layout_root,null);
```

Luego, llamamos al método "findViewById()", pero del objeto View generado, para buscar dentro de él, el objeto TextView contenido, al cual setearemos un texto.

Por último, creamos el objeto Toast, con el constructor que solo recibe el Context, luego seteamos la duración con el método "setDuration()" y la View, mediante "setView()".

El resultado de la ejecución del Toast es la siguiente:



SnackBar

A partir de Android 5.0 se incluyó un diálogo informativo que a diferencia del *Toast* puede tener interacción con el usuario. Para utilizar este diálogo en cualquier versión de Android, deberemos incluir la biblioteca *support design* en nuestro archivo *build.gradle* del módulo *app*.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.2.0'
    compile 'com.android.support:design:22.2.0'
}
```

A diferencia del *Toast*, este diálogo debe ser definido en nuestro archivo XML de layout, mediante un *CoordinatorLayout*:

```
<android.support.design.widget.CoordinatorLayout
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:layout_alignParentBottom="true"
  android:layout_centerHorizontal="true"
  android:id="@+id/snackbarPosition">
</android.support.design.widget.CoordinatorLayout>
```

Otra diferencia con el *Toast*, es que al tener un botón (llamado "acción" del *Snackbar*) deberemos asignar un *listener* que implemente la *inteface* **OnClickListener** (la misma que para los objetos *Button*) en el siguiente ejemplo, implementaremos la *interface* en nuestra *Activity*:

```
public class MainActivity extends AppCompatActivity implements
                                                    View.OnClickListener {
```

Esto nos obligará a implementar el método *onClick*, donde recibiremos el evento si el usuario presiona el botón del *Snackbar*:

```
@Override
public void onClick(View v) {
      Log.d("Activity", "Click!");
}
```

Ahora crearemos el *Snackbar* y lo mostraremos en pantalla, para ello, necesitamos la referencia del *CoodinatorLayout* que se encuentra en nuestro *layout*:

```
final View coordinatorLayoutView = findViewById(R.id.snackbarPosition);
Snackbar snack = Snackbar.make(coordinatorLayoutView, "No hay conexión",
                                                      Snackbar.LENGTH_LONG);
snack.setAction("Reintentar", this);
snack.show();
```

Primero obtenemos la referencia del *CoordinatorLayout* y luego mediante el método estático *make* de la clase *Snackbar* creamos un objeto *Snackbar*, pasándole el mensaje y la duración.

De manera opcional, mediante *setAction*, podemos indicarle que aparezca un botón, indicando el texto y el *listener* para el mismo.

Mediante el método *show* la *Snackbar* aparecerá en pantalla: