Lesson 11

Android Dialog Boxes

AlertDialog & Toast Widgets

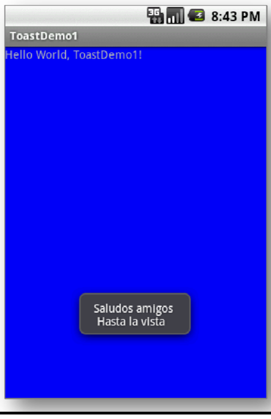
Victor Matos
Cleveland State University

Notes are based on:
Android Developers
http://developer.android.com/index.html

The DialogBox

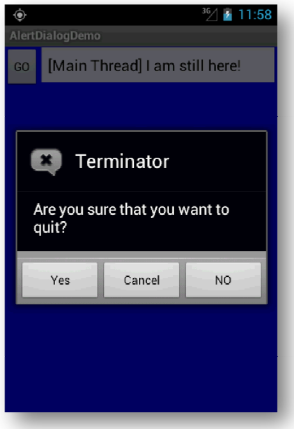
Android provides two primitive forms of dialog boxes:

1. **AlertDialog** boxes, and
2. **Toast** views



Hello World, ToastDemo1!

Saludos amigos
Hasta la vista



[Main Thread] I am still here!

Terminator

Are you sure that you want to quit?

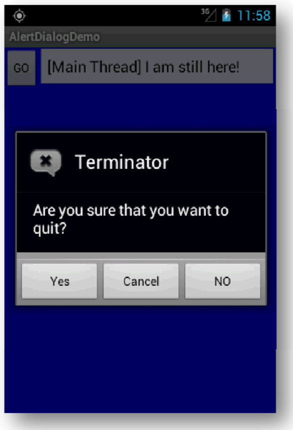
The AlertDialog

The **AlertDialog** is a simple message box that:

(1) Presents a brief message to the user

(2) Displays as a small floating window on top of the current UI

(3) Collects a simple answer (by clicking one of up to 3 buttons) .



[Main Thread] I am still here!

Terminator

Are you sure that you want to quit?

Note:
DialogBoxes are NOT modal views!
A fully *modal* view remains on the screen waiting for user's input. *The rest of the application is on hold*. It has to be dismissed by an explicit user's action.

The AlertDialog

Warning !

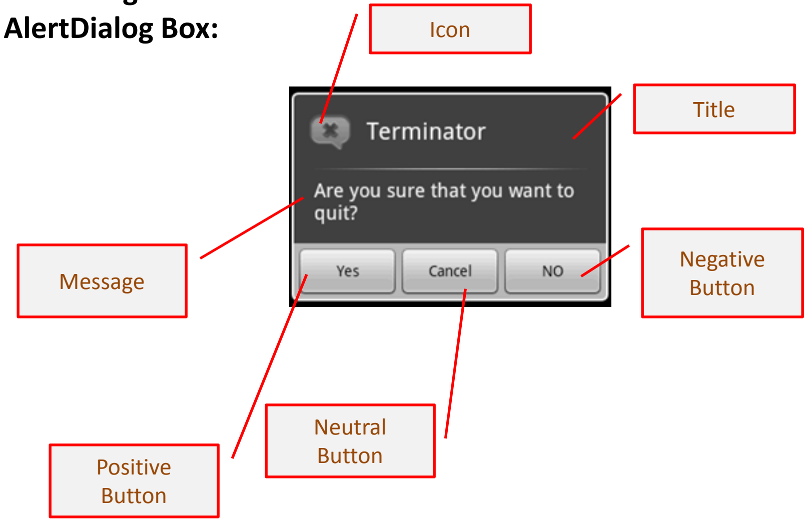
An *AlertDialog* is **NOT** a typical **synchronous** *inputBox* (as in .NET)

Why not?

Although *AlertDialog* boxes require user intervention to be terminated, they *do not stop the main thread.*

The AlertDialog

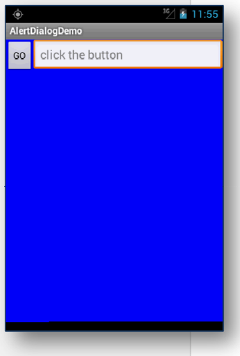
Dissecting an AlertDialog Box:



The AlertDialog

Example 1: Using a simple Dialog Box

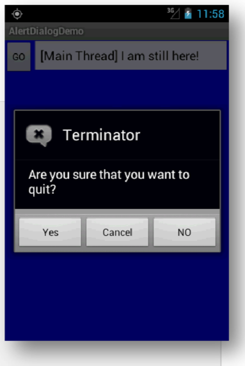
```
<LinearLayout
   android:id="@+id/LinearLayout01"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:background="#ff0000ff">

<Button
   android:text="GO"
   android:id="@+id/btnGo"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content">
 </Button>

 <EditText
   android:hint="click the button"
   android:id="@+id/txtMsg"
   android:layout_width="match_parent"
   android:layout_height="wrap_content">
 </EditText>

</LinearLayout>
```



The AlertDialog

Example 1: Using a simple dialog box

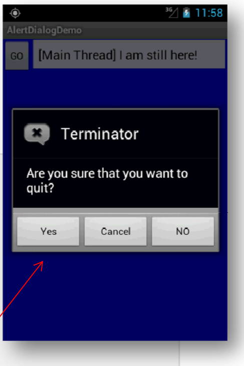
```
public class AlertDialogDemo extends Activity {
Button btnGo;
EditText txtMsg;
String msg;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg =  (EditText)findViewById(R.id.txtMsg);

        btnGo = (Button) findViewById(R.id.btnGo);
        btnGo.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View arg0) {

           AlertDialog dialogBox = makeAndShowDialogBox();
           dialogBox.show();

           // WARNING: (in general...) after showing a dialog you should have
           // NO more code. Let DialogBox's buttons handle the rest of the logic.
           txtMsg.setText("[Main Thread] I am still here!");
        }
        });

    }//onCreate
```


11:58
AlertDialogDemo
GO
[Main Thread] I am still here!
Terminator
Are you sure that you want to quit?
Yes
Cancel
NO

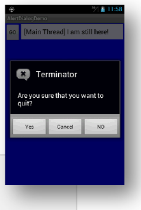
The AlertDialog

Example 1: Using a simple dialog box

```
private AlertDialog makeAndShowDialogBox(){

        AlertDialog myQuittingDialogBox =

        new AlertDialog.Builder(this)
           //set message, title, and icon
           .setTitle("Terminator")
           .setMessage("Are you sure that you want to quit?")
           .setIcon(R.drawable.ic_menu_end_conversation)

           //set three option buttons
           .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
              public void onClick(DialogInterface dialog, int whichButton) {
                 //whatever should be done when answering "YES" goes here
                 msg = "YES " + Integer.toString(whichButton);
                 txtMsg.setText(msg);
              }
           })//setPositiveButton
```


11:58
AlertDialogDemo
GO
[Main Thread] I am still here!
Terminator
Are you sure that you want to quit?
Yes
Cancel
NO

The AlertDialog

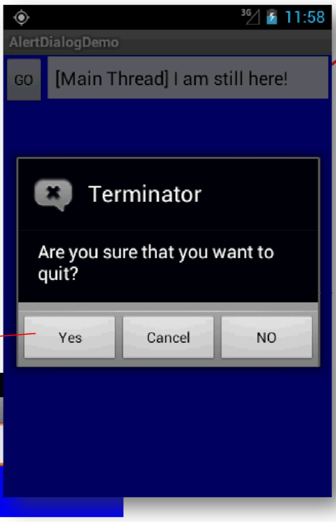


Example 1: Using a simple dialog box

```
      .setNeutralButton("Cancel", new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int whichButton) {
             //whatever should be done when answering "CANCEL" goes here
             msg = "CANCEL " + Integer.toString(whichButton);
             txtMsg.setText(msg);
          }//OnClick
       })//setNeutralButton

       .setNegativeButton("NO", new DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog, int whichButton) {
                //whatever should be done when answering "NO" goes here
             msg = "NO " + Integer.toString(whichButton);
             txtMsg.setText(msg);
            }
       })//setNegativeButton

       .create();

       return myQuittingDialogBox;
    }

}//AlertDialogDemo
```

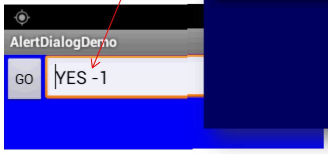
The AlertDialog

Example 1: Using a simple AlertDialog box


11:58
AlertDialogDemo
GO
[Main Thread] I am still here!
Terminator
Are you sure that you want to quit?
Yes
Cancel
NO

1. Background UI is obscured by DialogBox.
2. This text is set right after showing the dialog box
3. DialogBox remains on top until button is clicked.

After pressing "Yes" button

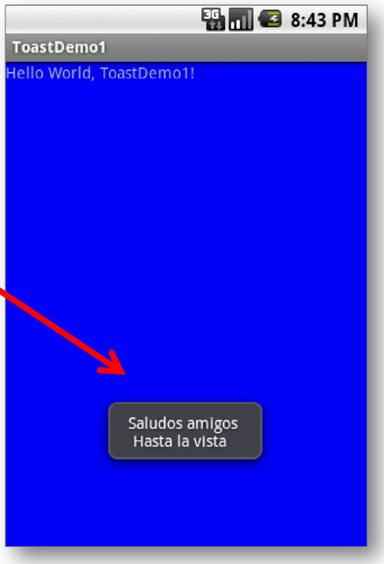


The Toast View

A **Toast** is a popup view that flashes –for a brief moment– a short message to the user.

They appear as a floating view over the application for 2-4 sec.

Toasts never receive focus !



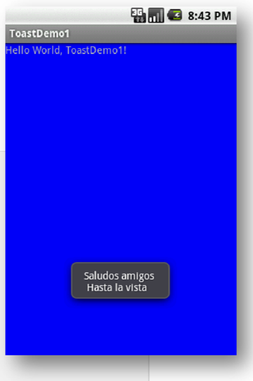
The Toast View

Syntax:

```
Toast.makeText ( context,  message,  duration ).show();
```

Context:	A reference to the view's environment (what is around me…)
Message:	The message you want to say
Duration:	Toast.*LENGTH_SHORT* (0) about 2 sec Toast.*LENGTH_* LONG (1) about 3.5 sec

The Toast View


8:43 PM
ToastDemo1
Hello World, ToastDemo1!
Saludos amigos
Hasta la vista

Example 2: A simple Toast

```
public class ToastDemo1 extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Toast.makeText(  getApplicationContext(),
                         "Saludos amigos \n Hasta la vista",
                         Toast.LENGTH_LONG).show();
    }
}
```

The Toast View

As an aside

Context:

On Android, a Context is mostly used to load and access resources.

All widgets receive a Context parameter in their constructor.

In a regular Android application, you usually have two kinds of Context, *Activity* and *Application*. The first one is typically passed to classes and methods that need a Context.

Views have a reference to the entire activity and therefore to anything your activity is holding onto; usually the entire View hierarchy and all its resources.

The Toast View

Customizing a Toast View

- By **default** Toast views are displayed at the **center-bottom** of the screen.
- However the user may change the placement of a Toast view by using either of the following methods:

```
void setGravity (int gravity, int xOffset, int yOffset)

void setMargin (float horizontalMargin, float verticalMargin)
```

The Toast View

Re-Positioning a Toast View – Method 1

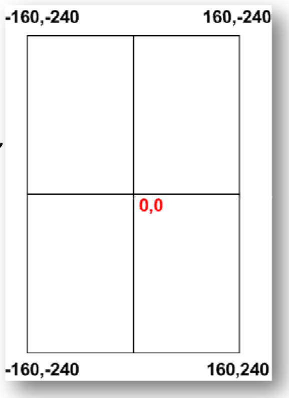
```
void setGravity (int gravity, int xOffset, int yOffset)
```

(Assume the phone has a 320x480 screen density)

Gravity: Overall placement. Typical values include: *Gravity.CENTER, Gravity.TOP, Gravity.BOTTOM, (see Apendix B)*

xOffset: The *xOffset* range is -160,…,0,…160
left center right

yOffset: The *yOffset* range is: -240,…,0,…240
top, center, bottom

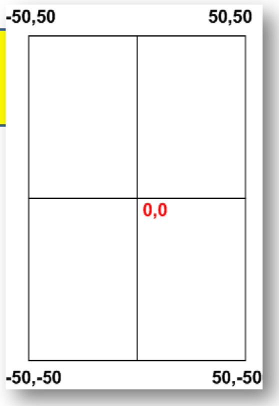


The Toast View

Re-Positioning the Toast View – Method 2

- The screen's center point is the where horizontal and vertical center lines meet.
- There is 50% of the screen to each side of that center point
- Margins are expressed as a value between: -50,…, 0, …, 50.

```
void setMargin (float horizontalMargin,
                float verticalMargin)
```

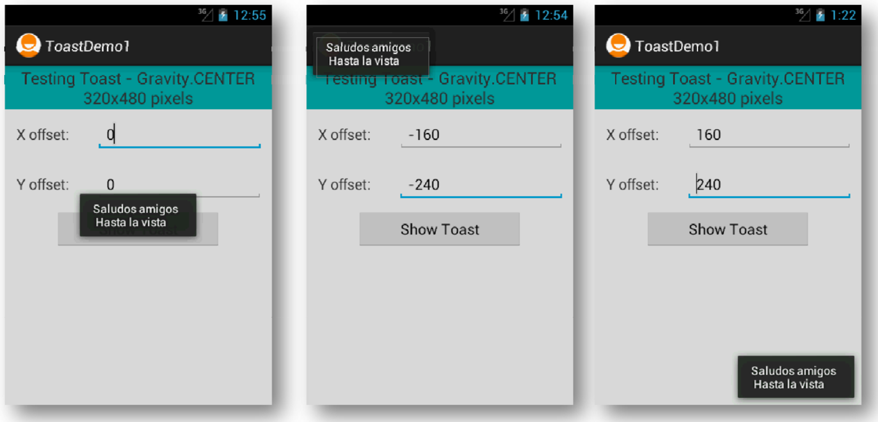


Note: The pair of margins:

(-50, -50)	represent the lower-left corner of the screen,
(0, 0)	is the center, and
(50, 50)	the upper-right corner.

The Toast View

Example 2: Changing the placement of a Toast view.

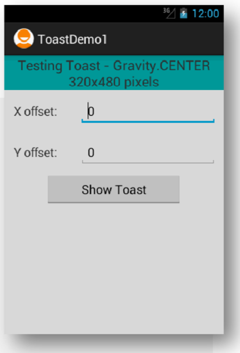


Using the **`setGravity(…)`** method with Gravity.CENTER, and x and y offsets of (resp.):

0, 0	(center)	[assuming device's density to be 360x480]
-160, -240	(top-left)	
160, 240	(right-bottom)	

The Toast View

Example2: Changing the placement of a Toast view. (main.xml 1 of 3)

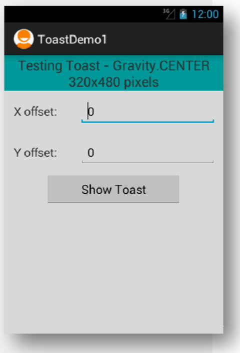


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ffdddddd"
    android:orientation="vertical" >

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#ff009999"
        android:gravity="center"
        android:text="Testing Toast - Gravity.CENTER  320x480 pixels"
        android:textSize="20sp" >
    </TextView>
```

The Toast View

Example2: Changing the placement of a Toast view. (main.xml 2 of 3)

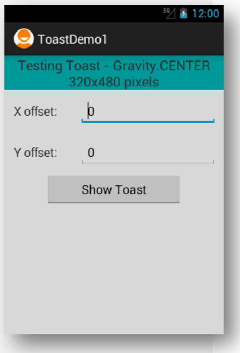
```
 <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10px" >

        <TextView
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:text=" X offset: "
            android:textSize="18sp" >
        </TextView>

        <EditText
            android:id="@+id/xBox"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:inputType="numberSigned"
            android:text="0"
            android:textSize="18sp" />
    </LinearLayout>
```



The Toast View

Example2: Changing the placement of a Toast view. (main.xml 3 of 3)

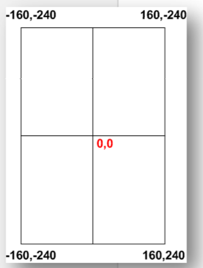
```
 <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10px" >

        <TextView
            android:layout_width="100dp"
            android:layout_height="wrap_content"
            android:text=" Y offset: "
            android:textSize="18sp"  />

        <EditText
            android:id="@+id/yBox"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:inputType="numberSigned"
            android:text="0"
            android:textSize="18sp"  />
</LinearLayout>

    <Button
        android:id="@+id/btn1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text=" Show Toast " >
    </Button>

</LinearLayout>
```



The Toast View

Example2: Changing the placement of a Toast view (assume 360x480)

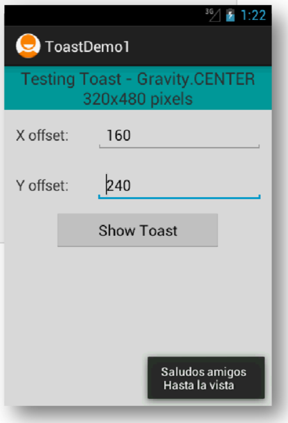
```
public class ToastDemo1 extends Activity {
    EditText xBox;
    EditText yBox;
    Button   btn1;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        xBox = (EditText)findViewById(R.id.xBox);
        yBox = (EditText)findViewById(R.id.yBox);
        btn1 = (Button)findViewById(R.id.btn1);
        btn1.setOnClickListener(new OnClickListener() {

          @Override
          public void onClick(View v) {
               try {
                 Toast myToast =  Toast.makeText(
                        getApplicationContext(),
                        "Saludos amigos \n Hasta la vista",
                        Toast.LENGTH_LONG);
                 myToast.setGravity(Gravity.CENTER,
                        Integer.valueOf(xBox.getText().toString()),
                        Integer.valueOf(yBox.getText().toString()) );

                 myToast.show();
```



The Toast View

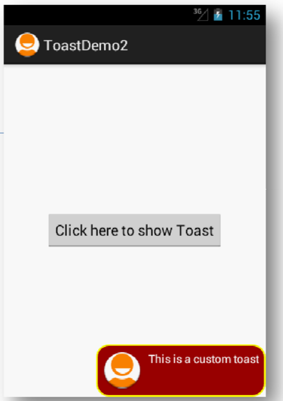
Example2: Changing the placement of a Toast view.

```
            } catch (NumberFormatException e) {
               Toast.makeText(getApplicationContext(),
                      e.getMessage(),
                      Toast.LENGTH_LONG).show();
            }
         }
      });

   }//onCreate

}//class
```



The Toast View

Example 3: Showing Custom-Made Toast Views



Toasts could be modified to display a custom combination of color, shape, text, image, and background.

To create a custom Toast follow the next steps:

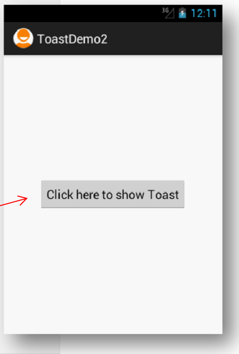
1. Define the XML layout of the new Toast custom view
2. Make sure there is a *TextView* named: **`text`**
3. Additionally you could attach an android: background to the TextView.
4. The background could be a figure (such as a *png* file) or an XML defined shape (see next example and Appendix B).

Example based on:
http://hustleplay.wordpress.com/2009/07/23/replicating-default-android-toast/
http://developer.android.com/guide/topics/ui/notifiers/toasts.html

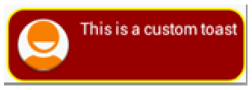
The Toast View

Example 3: Showing Custom-Made Toast Views.
Let’s begin with the application’s main layout.

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="Click here to show Toast"
        android:onClick="showCustomToast"
        tools:context=".ToastDemo2" />

</RelativeLayout>
```



The Toast View



Example 3: Showing Custom-Made Toast Views
Now we create our custom Toast layout called: *my_toast_layout.xml*.
It must contain a TextView called: *text*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:id="@+id/toast_layout_root"
              android:orientation="horizontal"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:padding="8dp"
              android:background="@layout/my_shape"
              >
    <ImageView android:src="@drawable/ic_launcher"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:layout_marginRight="8dp"
               />
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:textColor="#FFF"
              />
</LinearLayout>
```

Optional background image or shape

The Toast View

Example 3: Showing Custom-Made Toast Views (see appendix B)

- Finally we take care of the optional **background** element (*my_border.xml*).
- In this example we defined a <shape> element (it could also be any png image).
- Our XML shape file is saved in the folder: /res/layout

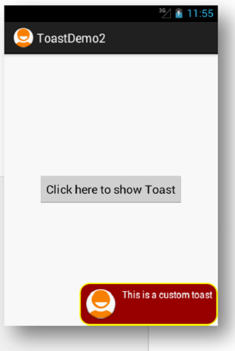
```xml
<?xml version="1.0" encoding="UTF-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle" >

    <stroke
        android:width="2dp"
        android:color="#ffffff00" />

    <solid android:color="#ff990000" />

    <padding
        android:bottom="4dp"
        android:left="10dp"
        android:right="10dp"
        android:top="4dp" />

    <corners android:radius="15dp" />

</shape>
```

Reddish rectangle ,
yellow edges,
Rounded corners

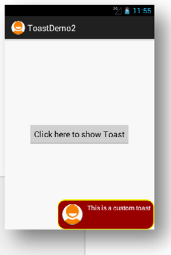
The Toast View

Example 3: Showing Custom-Made Toast Views

```java
public class ToastDemo2 extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

    }//onCreate

    public void showCustomToast(View v){
      // ///////////////////////////////////////////////
      // this fragment creates a custom Toast showing
      // image + text + shaped_background
      // triggered by XML button's android:onClick=...

      Toast customToast = makeCustomToast();
      customToast.show();

    } //showCustomToast
```



The Toast View



Example 3: Showing Custom-Made Toast Views

```
protected Toast makeCustomToast() {
// Reference:
// http://developer.android.com/guide/topics/ui/notifiers/toasts.html

   LayoutInflater inflater = getLayoutInflater();
   View layout = inflater.inflate(
                  R.layout.custom_toast,
                  (ViewGroup) findViewById(R.id.toast_layout_root));

   TextView text = (TextView) layout.findViewById(R.id.text);
   text.setText("This is a custom toast");

   Toast toast = new Toast(getApplicationContext());
   toast.setMargin(50,-50); //lower-right corner
   toast.setDuration(Toast.LENGTH_LONG);
   toast.setView(layout);

   return toast;
}//makeCustomToast

}//ToastDemo2
```

The Toast View

Comment

Inflating a View

- Once the Hierarchy View has been displayed, you can take any terminal node and **extend it** by inflating a custom ‘*view sub-tree*’.
- Also, by using layout inflation we may draw a new Hierarchy on top of the existing screen.

The Toast View

Comment: Inflating a Toast View

Syntax

`public View inflate (int resource, ViewGroup root)`

Inflate a new view hierarchy from the specified xml resource.

Parameters

resource ID for an XML layout resource to load,

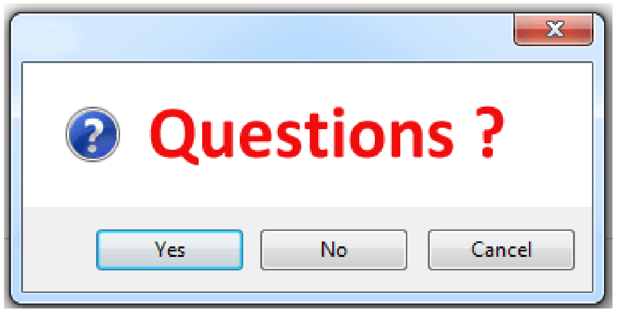
root: optional view to be the parent of the generated hierarchy.

Returns

The root View of the inflated hierarchy. If root was supplied, this is the root View; otherwise it is the root of the inflated XML file.

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(
                  R.layout.my_toast_layout,
                  (ViewGroup) findViewById(R.id.my_toast_layout_root));
TextView text = (TextView) layout.findViewById(R.id.text);
```

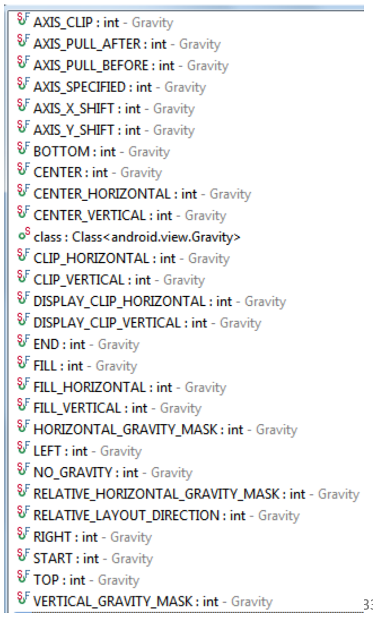
Dialog Boxes



Dialog Boxes

Appendix A.

Standard Gravity Values



```
AXIS_CLIP : int - Gravity
AXIS_PULL_AFTER : int - Gravity
AXIS_PULL_BEFORE : int - Gravity
AXIS_SPECIFIED : int - Gravity
AXIS_X_SHIFT : int - Gravity
AXIS_Y_SHIFT : int - Gravity
BOTTOM : int - Gravity
CENTER : int - Gravity
CENTER_HORIZONTAL : int - Gravity
CENTER_VERTICAL : int - Gravity
class : Class<android.view.Gravity>
CLIP_HORIZONTAL : int - Gravity
CLIP_VERTICAL : int - Gravity
DISPLAY_CLIP_HORIZONTAL : int - Gravity
DISPLAY_CLIP_VERTICAL : int - Gravity
END : int - Gravity
FILL : int - Gravity
FILL_HORIZONTAL : int - Gravity
FILL_VERTICAL : int - Gravity
HORIZONTAL_GRAVITY_MASK : int - Gravity
LEFT : int - Gravity
NO_GRAVITY : int - Gravity
RELATIVE_HORIZONTAL_GRAVITY_MASK : int - Gravity
RELATIVE_LAYOUT_DIRECTION : int - Gravity
RIGHT : int - Gravity
START : int - Gravity
TOP : int - Gravity
VERTICAL_GRAVITY_MASK : int - Gravity
```

Dialog Boxes

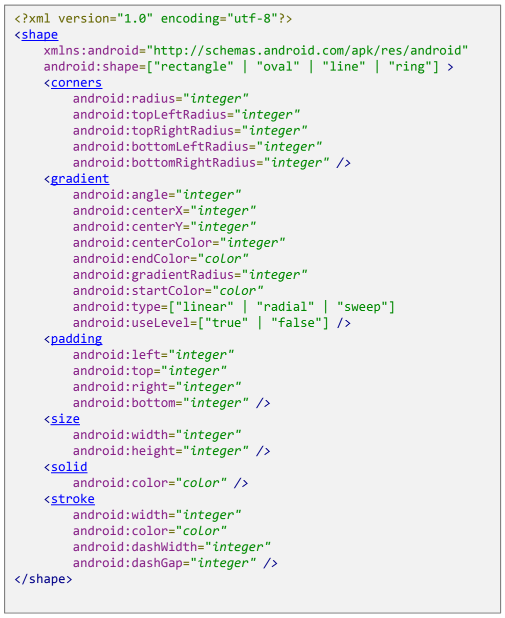
Appendix B.
Shape Drawable

It is an XML file that defines a geometric shape, including colors and gradients.

Some basic shapes are:
rectangle, oval, ring, line

References:
http://developer.android.com/reference/android/graphics/drawable/shapes/Shape.html

http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape



```xml
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape=["rectangle" | "oval" | "line" | "ring"] >
    <corners
        android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer" />
    <gradient
        android:angle="integer"
        android:centerX="integer"
        android:centerY="integer"
        android:centerColor="integer"
        android:endColor="color"
        android:gradientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" | "sweep"]
        android:useLevel=["true" | "false"] />
    <padding
        android:left="integer"
        android:top="integer"
        android:right="integer"
        android:bottom="integer" />
    <size
        android:width="integer"
        android:height="integer" />
    <solid
        android:color="color" />
    <stroke
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer" />
</shape>
```