Android - Alert Dialog Tutorial

A Dialog is small window that prompts the user to a decision or enter additional information.

Some times in your application, if you wanted to ask the user about taking a decision between yes or no in response of any particular action taken by the user, by remaining in the same activity and without changing the screen, you can use Alert Dialog.

In order to make an alert dialog, you need to make an object of AlertDialogBuilder which an inner class of AlertDialog. Its syntax is given below

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

Now you have to set the positive (yes) or negative (no) button using the object of the AlertDialogBuilder class. Its syntax is

alertDialogBuilder.setPositiveButton(CharSequence text, DialogInterface.OnClickListener listener)

alertDialogBuilder.setNegativeButton(CharSequence text, DialogInterface.OnClickListener listener)

Apart from this , you can use other functions provided by the builder class to customize the alert dialog. These are listed below

|  |  |
| --- | --- |
| **Sr.No** | **Method type & description** |
| 1 | **setIcon(Drawable icon)**  This method set the icon of the alert dialog box. |
| 2 | **setCancelable(boolean cancel able)**  This method sets the property that the dialog can be cancelled or not |
| 3 | **setMessage(CharSequence message)**  This method sets the message to be displayed in the alert dialog |
| 4 | **setMultiChoiceItems(CharSequence[] items, boolean[] checkedItems, DialogInterface.OnMultiChoiceClickListener listener)**  This method sets list of items to be displayed in the dialog as the content. The selected option will be notified by the listener |
| 5 | **setOnCancelListener(DialogInterface.OnCancelListener onCancelListener)**  This method Sets the callback that will be called if the dialog is cancelled. |
| 6 | **setTitle(CharSequence title)**  This method set the title to be appear in the dialog |

After creating and setting the dialog builder , you will create an alert dialog by calling the create() method of the builder class. Its syntax is

AlertDialog alertDialog = alertDialogBuilder.create();

alertDialog.show();

This will create the alert dialog and will show it on the screen.

Dialog fragment

Before enter into an example we should need to know dialog fragment.Dialog fragment is a fragment which can show fragment in dialog box

public class DialogFragment extends DialogFragment {

@Override

public Dialog onCreateDialog(Bundle savedInstanceState) {

// Use the Builder class for convenient dialog construction

AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

builder.setPositiveButton(R.string.fire, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

toast.makeText(this,"enter a text here",Toast.LENTH\_SHORT).show();

}

})

.setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int id) {

finish();

});

// Create the AlertDialog object and return it

return builder.create();

}

}

}

List dialog

It has used to show list of items in a dialog box.For suppose, user need to select a list of items or else need to click a item from multiple list of items.At this situation we can use list dialog.

public Dialog onCreateDialog(Bundle savedInstanceState) {

AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

builder.setTitle(Pick a Color)

.setItems(R.array.colors\_array, new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

// The 'which' argument contains the index position

// of the selected item

}

});

return builder.create();

}

Single-choice list dialog

It has used to add single choice list to Dialog box.We can check or uncheck as per user choice.

public Dialog onCreateDialog(Bundle savedInstanceState) {

mSelectedItems = new ArrayList();

AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

builder.setTitle("This is list choice dialog box");

.setMultiChoiceItems(R.array.toppings, null,new DialogInterface.OnMultiChoiceClickListener() {

@Override

public void onClick(DialogInterface dialog, int which, boolean isChecked) {

if (isChecked) {

// If the user checked the item, add it to the selected items

mSelectedItems.add(which);

}

else if (mSelectedItems.contains(which)) {

// Else, if the item is already in the array, remove it

mSelectedItems.remove(Integer.valueOf(which));

}

}

})

// Set the action buttons

.setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int id) {

// User clicked OK, so save the mSelectedItems results somewhere

// or return them to the component that opened the dialog

...

}

})

.setNegativeButton(R.string.cancel, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int id) {

...

}

});

return builder.create();

}

Example

The following example demonstrates the use of AlertDialog in android.

To experiment with this example , you need to run this on an emulator or an actual device.

|  |  |
| --- | --- |
| **Steps** | **Description** |
| 1 | You will use Android studio to create an Android application and name it as My Application under a package package com.example.sairamkrishna.myapplication; While creating this project, make sure you Target SDK and Compile With at the latest version of Android SDK to use higher levels of APIs. |
| 2 | Modify src/MainActivity.java file to add alert dialog code to launch the dialog. |
| 3 | Modify layout XML file res/layout/activity\_main.xml add any GUI component if required. |
| 4 | No need to change default string constants. Android studio takes care of default strings at values/string.xml |
| 9 | Run the application and choose a running android device and install the application on it and verify the results. |

Here is the modified code of **src/MainActivity.java**

package com.example.sairamkrishna.myapplication;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.support.v7.app.ActionBarActivity;

import android.os.Bundle;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.Toast;

public class MainActivity extends ActionBarActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

public void open(View view){

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

alertDialogBuilder.setMessage("Are you sure,You wanted to make decision");

alertDialogBuilder.setPositiveButton("yes", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface arg0, int arg1) {

Toast.makeText(MainActivity.this,"You clicked yes button",Toast.LENGTH\_LONG).show();

}

});

alertDialogBuilder.setNegativeButton("No",new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

finish();

}

});

AlertDialog alertDialog = alertDialogBuilder.create();

alertDialog.show();

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu\_main, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

//noinspection SimplifiableIfStatement

if (id == R.id.action\_settings) {

return true;

}

return super.onOptionsItemSelected(item);

}

}

Here is the modified code of **res/layout/activity\_main.xml**

<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" android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:paddingBottom="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Alert Dialog"

android:id="@+id/textView"

android:textSize="35dp"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Tutorialspoint"

android:id="@+id/textView2"

android:textColor="#ff3eff0f"

android:textSize="35dp"

android:layout\_below="@+id/textView"

android:layout\_centerHorizontal="true" />

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/imageView"

android:src="@drawable/logo"

android:layout\_below="@+id/textView2"

android:layout\_alignRight="@+id/textView2"

android:layout\_alignEnd="@+id/textView2"

android:layout\_alignLeft="@+id/textView"

android:layout\_alignStart="@+id/textView" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Alert dialog"

android:id="@+id/button"

android:layout\_below="@+id/imageView"

android:layout\_alignRight="@+id/textView2"

android:layout\_alignEnd="@+id/textView2"

android:layout\_marginTop="42dp"

android:onClick="open"

android:layout\_alignLeft="@+id/imageView"

android:layout\_alignStart="@+id/imageView" />

</RelativeLayout>

Here is of**Strings.xml**

<resources>

<string name="app\_name">My Application</string>

<string name="hello\_world">Hello world!</string>

<string name="action\_settings">Settings</string>

</resources>

Here is the default code of **AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.alertdialog"

android:versionCode="1"

android:versionName="1.0" >

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name="com.example.sairamkrishna.myapplication.MainActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

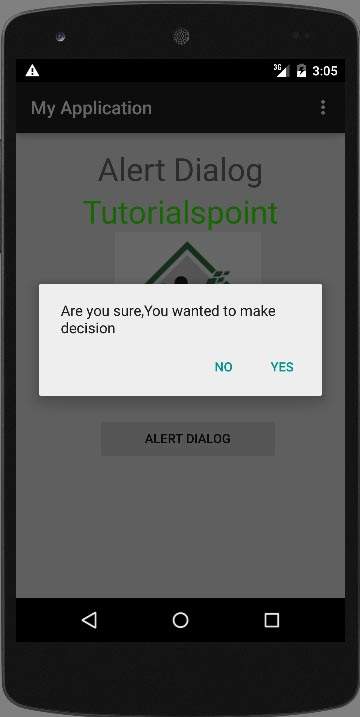
</intent-filter>

</activity>

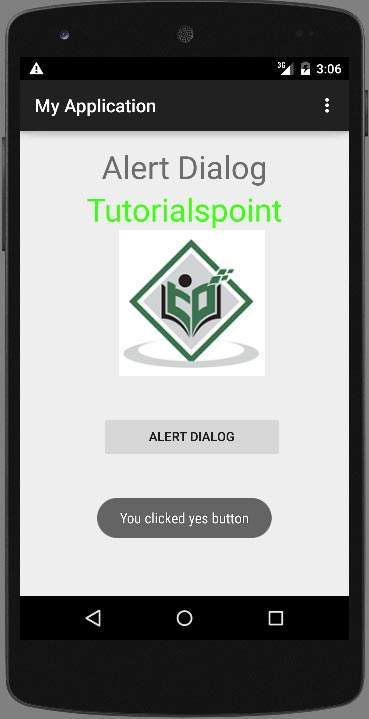
</application>

</manifest>

Let's try to run your application. I assume you have connected your actual Android Mobile device with your computer. To run the app from Android studio, open one of your project's activity files and click Run Eclipse Run Icon icon from the toolbar. Before starting your application, ]Android studio will display following window to select an option where you want to run your Android application.



Select your an option and then click on it. For suppose, if you have clicked on yes button, then result would as follows



if you click on no button it will call finish() and it will close your application.