- Dream.In.Code> Programming Tutorials
- > Mobile Development Tutorials> Android DevelopmentPage 1 of 1

Android, Part III: Dynamic Layouts

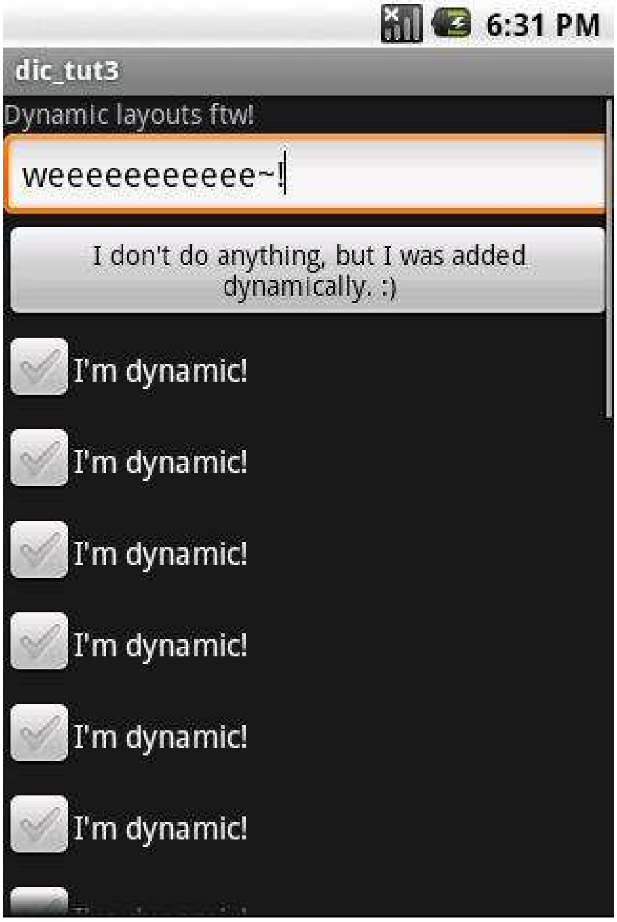
gabehabe
Posted 07 October 2009 - 11:38 AM

POPULAR

Dynamic Layouts in Android

No XML in this tutorial! We'll be completely designing our application in Java, and creating our layout completely dynamically. The benefits of this are that we can add an undefined amount of widgets at run time. This is a pretty useful skill, but not brilliantly documented. (Everything is ZOMG LAYOUT.XML FTW!!1!)

We're only going to create a crude, eventless UI, but it's a good learning exercise:



6:31 PM

dic_tut3

Dynamic layouts ftw!

weeeeeeeeeee~!

I don't do anything, but I was added dynamically. :)

☐ I'm dynamic!

☐ I'm dynamic!

☐ I'm dynamic!

☐ I'm dynamic!

☐ I'm dynamic!

☐ I'm dynamic!

Okay, so let's get started. We won't be touching on events in this tutorial, our app won't actually *do* anything, per se... it's merely an example of adding stuff dynamically. You'll notice how we set the properties through Java, quite similarly to how we define them in the XML.

To start off, remember we're creating a package, as usual.
Resized to 96% (was 525 x 686) - Click image to enlarge

Follow & Share

Follow

BY FEEDBURNER

Tutorials

Programming

- C++ Tutorials
- Java Tutorials
- VisualBasic Tutorials
- VB.NET Tutorials
- C# Tutorials
- PHP Tutorials
- ColdFusion Tutorials
- Database Tutorials

Web Development

- HTML Tutorials
- JavaScript Tutorials
- CSS Tutorials
- Flash Tutorials
- Photoshop Tutorials

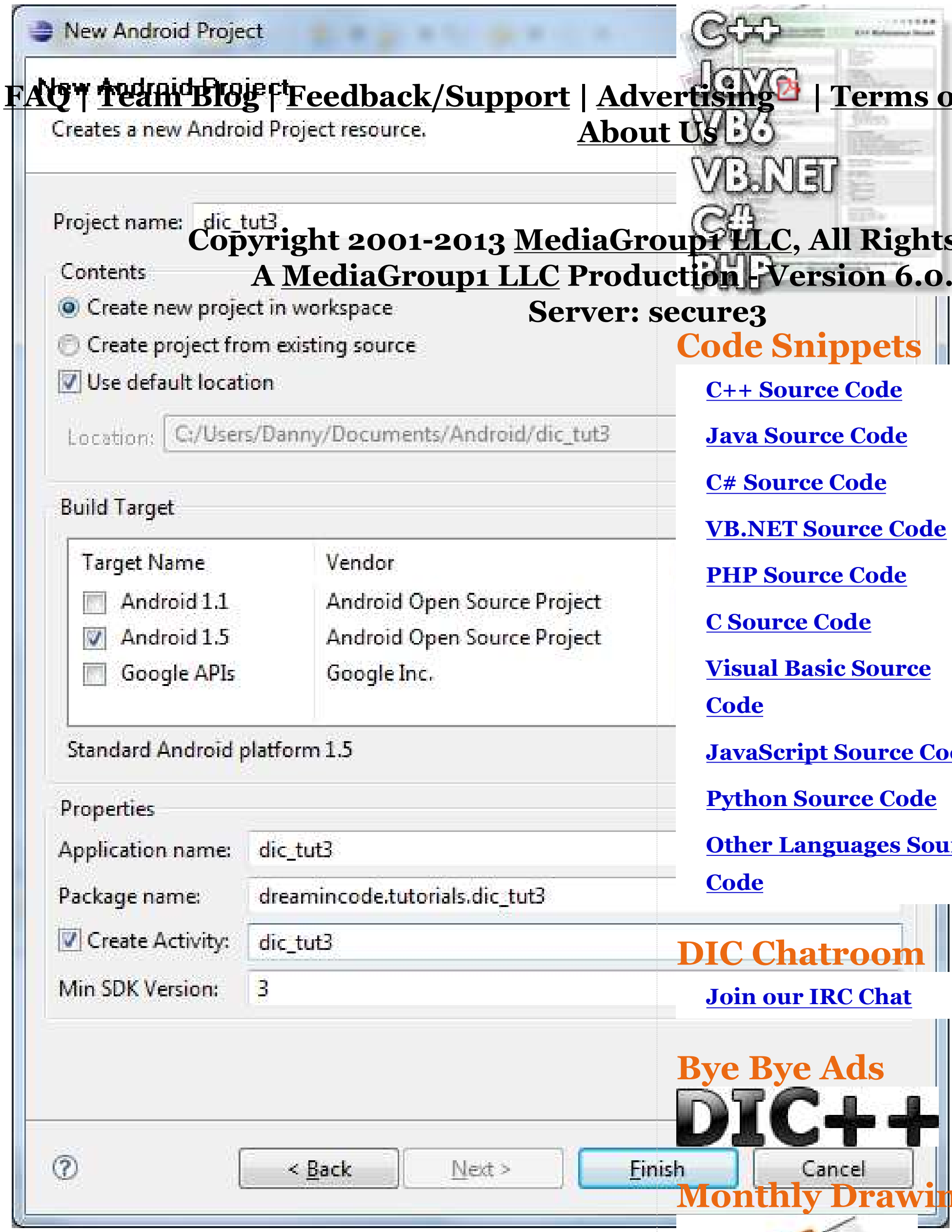
Reference Sheets

General Discussion

- Caffeine Lounge
- Corner Cubicle
- Student Campus
- Software Development
- Industry News
- Introduce Yourself
- Nightmare.In.Code

Programming Help

- C and C++
- VB.NET
- Java
- C#
- ASP.NET
- .NET Framework
- VB6
- PHP
- Ruby
- Python
- ColdFusion
- Databases
- Other Languages
- Game Development
- Mobile Development
- 52 Weeks Of Code



New Android Project

Creates a new Android Project resource.

Project name: dic_tut3

Contents

● Create new project in workspace

○ Create project from existing source

☑ Use default location

Location: C:/Users/Danny/Documents/Android/dic_tut3

Build Target

Target Name	Vendor
□ Android 1.1	Android Open Source Project
☑ Android 1.5	Android Open Source Project
□ Google APIs	Google Inc.

Standard Android platform 1.5

Properties

Application name: dic_tut3

Package name: dreamincode.tutorials.dic_tut3

☑ Create Activity: dic_tut3

Min SDK Version: 3

```
package dreamincode.tutorials.dic_tut3;
```

Next, we're going to import the usual suspects... Activity and Bundle.

```
import android.app.Activity;
import android.os.Bundle;
```

The next two are new. One you may recognise from our layouts before... LinearLayout. We're also going to use a ScrollView. The reason we'll be using a ScrollView is that we're going to add a *hell* of a lot of widgets, and they wouldn't all fit on screen. If we don't put our LinearLayout inside a ScrollView, it simply won't scroll and the user won't be able to reach all the widgets!

```
import android.widget.ScrollView;
import android.widget.LinearLayout;
```

The last things we need to import are our widgets. We'll be using an EditText, a TextView, a Button and a CheckBox. We've already seen the first three in parts 1 and 2 of this tutorial, but the CheckBox is new. (And I hope the name is self-explanatory for what it is!) ;)

```
import android.widget.Button;
```

FAQ | Team Blog | Feedback/Support | Advertising | Terms of Use | Privacy Policy | About Us

Code Snippets

C++ Source Code

Java Source Code

C# Source Code

VB.NET Source Code

PHP Source Code

C Source Code

Visual Basic Source Code

JavaScript Source Code

Python Source Code

Other Languages Source Code

DIC Chatroom

Join our IRC Chat

Bye Bye Ads

Monthly Drawing

Web Development

Web Development

HTML & CSS

JavaScript

Graphic Design

Flash & ActionScript

Blogging

SEO & Advertising

Web Servers & Hosting

Site Check

```
import android.widget.TextView;
import android.widget.EditText;
import android.widget.CheckBox;
```

Now it's time to get on with our code. We're only going to extend Activity this time, remember we won't be using events. This bit's nothing new, so I'll just throw in the opening of the class and the opening of the onCreate() method. (Note, if you're not familiar with this, check part 1 (http://www.dreamincode.net/forums/index.php?showtopic=130052) and part 2 (http://www.dreamincode.net/forums/index.php?showtopic=130264))

```
public class dic_tut3 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

Time to get dirty. Dynamic layouts aren't always the most elegant of solutions, but they can be fun, and sometimes easier to implement. They're also *really* useful for certain occasions (such as SmartCalc*, my app: it uses a dynamic layout to decide how many textboxes are needed at runtime)
* No link, it's a paid app and I don't want to look like an ad whore. :)

Code first, explanation afterwards.

```
1 ScrollView sv = new ScrollView(this);
2 LinearLayout ll = new LinearLayout(this);
3 ll.setOrientation(LinearLayout.VERTICAL);
4 sv.addView(ll);
```

LINE 1: Creates a ScrollView object, which is a part of **this** Activity.
LINE 2: Creates a LinearLayout object, which you hopefully remember from the layout.xml we discussed before.
LINE 3: Sets the orientation of our LinearLayout. Remember how we used android:orientation="vertical" before? Same principal applies here.
LINE 4: Adds the LinearLayout to the ScrollView. Remember, we want to be able to Scroll the LinearLayout View.
NOTE: A ScrollView can **only** have one view added to it. If you try to add more than one, your application **WILL** force close. If you need to reset it, use ScrollView.removeAllViews(); and re-add whatever you need. (This is why we're going to add all our widgets to the LinearLayout as opposed to the ScrollView!)

Adding widgets dynamically
Actually, this same principal applies to all our widgets. But I'll use a few, just to demonstrate. :) This one should look relatively familiar, since we played with TextView.setText() in part 2.

```
TextView tv = new TextView(this);
tv.setText("Dynamic layouts ftw!");
ll.addView(tv);
```

Notice how we use addView() on ll, our LinearLayout, just like we added ll to sv? That's all it takes! :)

Now, let's do a Button and a EditText. Same principal, just like I said:

```
EditText et = new EditText(this);
et.setText("weeeeeeeeeee~!");
ll.addView(et);

Button b = new Button(this);
b.setText("I don't do anything, but I was added dynamically. :)");
ll.addView(b);
```

(Remember, when you run the application, the button won't actually do anything: We haven't added an onclickListener! If you want, you may want to revisit the button at the end of this tutorial and try adding an event by yourself. (Use part 2 of this series as a reference if you get stuck)

The next is still the same principal, but I've added it in a loop. 20 pointless CheckBoxes, coming up!

```
for(int i = 0; i < 20; i++) {
    CheckBox cb = new CheckBox(this);
    cb.setText("I'm dynamic!");
    ll.addView(cb);
}
```

This is really just to pad out our layout, so we can really see how useful that ScrollView is. :)

The very last thing we need to do is use setContentView() -- similar to how we did before. Remember the this.setContentView(R.layout.main);? Pretty much the same, only now we're going to use the ScrollView we created earlier instead:

```
this.setContentView(sv);
```

And lastly, close off the method and class!

```
    }
}
```

Now, if you run this, you'll get a rather crude, but biiiig layout, complete with all the widgets that we added dynamically.

Complete code:

```
package dreamincode.tutorials.dic_tut3;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ScrollView;
import android.widget.LinearLayout;
import android.widget.Button;
import android.widget.TextView;
import android.widget.EditText;
import android.widget.CheckBox;

public class dic_tut3 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ScrollView sv = new ScrollView(this);
        LinearLayout ll = new LinearLayout(this);
        ll.setOrientation(LinearLayout.VERTICAL);
        sv.addView(ll);

        TextView tv = new TextView(this);
        tv.setText("Dynamic layouts ftw!");
        ll.addView(tv);

        EditText et = new EditText(this);
        et.setText("weeeeeeeeeee~!");
        ll.addView(et);

        Button b = new Button(this);
        b.setText("I don't do anything, but I was added dynamically. :)");
        ll.addView(b);

        for(int i = 0; i < 20; i++) {
            CheckBox cb = new CheckBox(this);
            cb.setText("I'm dynamic!");
            ll.addView(cb);
        }
        this.setContentView(sv);
    }
}
```

See if you can add an onclickListener to the button, just have a play about with it until you feel fairly confident... I won't be explaining too much about the layouts in the coming tutorials, more on actual features such as SQLite. :)

Complete project: (http://www.dreamincode.net/forums/index.php?app=core&module=attach&section=attach&attach_id=14242&s=074a9b9e05ea26628b3f825d5b89906f) dic_tut3.zip (http://www.dreamincode.net/forums/index.php?app=core&module=attach&section=attach&attach_id=14242&s=074a9b9e05ea26628b3f825d5b89906f)
(25.83K)
Number of downloads: 5121

Replies To: Android, Part III: Dynamic Layouts

Guest_dalf*

Posted 11 March 2010 - 10:55 AM

Thanks for this easy to understand topic.
How can we do if we want to use XML layout?
For example, there is one file containing the scrollView only.
And another XML file containing 1 button only.

You want to add 20 buttons in the scrollView dynamically. Is it possible ?

Dalf

Guest_tommy*

Posted 26 May 2010 - 05:43 PM

> (http://www.dreamincode.net/forums/index.php?app=forums&module=forums&section=findpost&pid=955987) dalf, on 11 March 2010 - 09:55 AM, said:
> Thanks for this easy to understand topic.
> How can we do if we want to use XML layout?
> For example, there is one file containing the scrollView only.
> And another XML file containing 1 button only.
>
> You want to add 20 buttons in the scrollView dynamically. Is it possible ?
>
> Dalf

derefence the layout you want and make it find the name you gave in the xml file
(RelativeLayout) findViewById(R.id.RelativeLayout01);

same with the button but if you make 20 buttons, just drop the xml file and write
Button bt0 = new Button(this);
bt0.setId(0);
Button bt1 = new Button (this);
bt.setId(1);

don't forget to add the buttons to the view...

Guest_Sara*
Posted 24 August 2010 - 01:20 AM

It was a veryyyyyyyyyyyyy Useful one. Thank you :)

Guest_Paul*
Posted 25 September 2010 - 03:10 PM

Hi

I am trying to create a layout within another layout but I cannot get it to work.
I want one layout for standards buttons side by side and one for a custom view below the buttons.
Is there a command to add a layout to a layout. I tried addView but it does not work.
Is there an addLayout function somewhere?

Thanks
Paul

```
TableLayout layout = new TableLayout(this);
layout.setLayoutParams( new TableLayout.LayoutParams(1,2));

TableLayout layout2 = new TableLayout(this);
layout2.setLayoutParams( new TableLayout.LayoutParams(2,1));

layout2.addView(BtnClose, 0);
layout2.addView(BtnReset, 1);

layout.addView(mCustomDrawableView, 0);
layout.addView(layout2 ,1);  // This is the part that does not work
```

Guest_Mike W*
Posted 26 September 2010 - 06:42 PM

Gabehabe,

thanks a lot for the helpful tutorial!

I have implemented one layout based on your instructions, works great in portrait mode, however, everytime I go to landscape, it crashes...

Any ideas?

Guest_Apoorvas*
Posted 15 November 2010 - 04:44 AM

Hey.. can you expand a little bit on this? I am currently trying to dynamically add widgets using an xml file. I am currently doing this:

ScrollView sv = new ScrollView(this);
LinearLayout RL = (LinearLayout) findViewById(R.id.linearLayout1);

// But in order for this to return a non-null value I have to do setContextView(R.layout.layout)

RL.setOrientation(LinearLayout.VERTICAL);
setContentView(null);
sv.addView(RL);
Button b = new Button(this);
b.setText("I don't do anything, but I was added dynamically. :)");
RL.addView(B);
this.setContentView(sv);

It keeps giving an error though. I would really appreciate some help!

Thanks,
AS

(http://www.dreamincode.net/forums/index.php?app=forums&module=forums&section=findpost&pid=1027397) tommy, on 26 May 2010 - 04:43 PM, said:

(http://www.dreamincode.net/forums/index.php?app=forums&module=forums&section=findpost&pid=955987) dalf, on 11 March 2010 - 09:55 AM, said:

Thanks for this easy to understand topic.
How can we do if we want to use XML layout?
For example, there is one file containing the scrollView only.
And another XML file containing 1 button only.

You want to add 20 buttons in the scrollView dynamically. Is it possible ?

Dalf

derefence the layout you want and make it find the name you gave in the xml file
(RelativeLayout) findViewById(R.id.RelativeLayout01);

same with the button but if you make 20 buttons, just drop the xml file and write
Button bt0 = new Button(this);
bt0.setId(0);
Button bt1 = new Button (this);
bt.setId(1);

don't forget to add the buttons to the view...

Guest_nathan*
Posted 13 January 2011 - 11:40 AM

I was wondering about this snippet:

```
1   for(int i = 0; i < 20; i++) {
2       CheckBox cb = new CheckBox(this);
3       cb.setText("I'm dynamic!");
4       ll.addView(cb);
5   }
```

Would this create 20 checkboxes with the exact same name?

Would it be helpful to add this to the loop?

```
cb.setID(i);



for(int i = 0; i < 20; i++) {
        CheckBox cb = new CheckBox(this);
            cb.setID(i);        // Add here
????????
        cb.setText("I'm dynamic!");
        ll.addView(cb);
    }
```

Also, could you instead reference each CheckBox as a child of the LinearLayout?

pseudo code example:

```
if ( ll.child(2) == isChecked())    {
                          //do something
                          }
```

I'll have to take the time to figure it out myself, but thought it would be food for though in this discussion.

I'll post what I find out.

-nathan

Guest_nathan*
Posted 13 January 2011 - 06:39 PM

I worked out a roundabout way to create all my CheckBoxes within a loop , using gabehabes example, and then be able to access the elements at a later time, using the setID().

(I was having visibility problems and nullpointer exceptions when I tried to access my CheckBoxes outside of the For loop.)

I hope this might be useful:

inside onCreate:

```
ScrollView sv = new ScrollView(this);
      ll = new LinearLayout(this);
      ll.setOrientation(LinearLayout.VERTICAL);
      sv.addView(ll);
```

```

   TextView tv = new TextView(this);
   tv.setText("Dynamic layouts rock the house!");
   ll.addView(tv);

   CheckBox cha[] = addCheckBoxes();

   this.setContentView(sv);

   cha[4].setText("haha i can access 4 !!");
```

outside onCreate:

```
public CheckBox[] addCheckBoxes()  {
  CheckBox ca[] = new CheckBox[20];
  for(int i = 0; i < 20; i++) {
    CheckBox cb = new CheckBox(this);
      cb.setId(i);
    cb.setText("I'm CheckBox #: " + cb.getId());
    ll.addView(cb);
    ca[i] = cb;
  }
  return ca;
}
```

Guest_nathan*

Posted 14 January 2011 - 01:56 PM

Well, I've tried to dynamically layout and adjust the size of another View type (EditText), and am failing miserably.
I have tried inside a ScrollView, and outside.

I am trying to restrict the size of the EditText to 2 characters, for a simple golf scorecard app.
I need to fit a lot of EditTexts on the screen at the same time.

EditTexts seem to want to fill the parent object all the way horizontally.

```
et.setFilters(new InputFilter[]{new InputFilter.LengthFilter(2)});
```

This code successfully restricts the user to 2 characters of input into the EditText, but the stupid EditText ALWAYS stretches all the way across the screen.

I figured EditTexts might inherit the method setWidth(int pixels) from TextView, but setWidth() does nothing.

It may not be entirely dynamic, but would it be possible to create ONE .xml layout template describing the EditText exactly the way I want, then dynamically create dozens of them?
(XML layouts seem to be more "powerful" than the available Java methods.)

I suppose an alternative would be to define a custom View class, based on an EditText, then dynamically create as many EditTexts as needed.

I hope this post provokes more thought.

scottbiggs

Posted 23 February 2011 - 04:30 PM

Excellent! Just what I was looking for and brilliantly clear. Thank you!

-scott

darkzadow

Posted 21 May 2011 - 04:45 PM

Hi i was wondering how do you add functionality to the button. Whatever i try to do it ends up force closing.

Dogstopper
Posted 21 May 2011 - 04:50 PM

You can do something like this:

```
button.setonclickListener(new View.onclickListener() {
             public void onclick(View v) {
                 // Perform action on click
             }
         });
```

If you need more help, feel free to post in the Android forum

darkzadow
Posted 21 May 2011 - 07:10 PM

Im using the
ScrollView sv = New ScrollView
ll = new LinearLayout(this);
Button b = new Button(this);
b.setText("Search");
b.setId(-2);
ll.addView(B);
as detailed by the above tutorial
I created a button exactly like it is shown.
When I try to make the button do anything i get a force close or have to caste view as (view)this.
Is there a special method or way to get the buttons created above to do anything?

master.bennett
Posted 04 February 2012 - 05:47 PM

I know this thread is old but felt no need to create a new topic.

I have been recently going through some android tutorials and just in case anyone wants to get this example using xml for the layout, here's the code below:

The Java activity file:

```
package org.pg;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

public class PlaygroundActivity extends Activity {

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }

  public void addName(View view) {
    String name = ((EditText) this.findViewById(R.id.txt_name)).getText().toStr

    TextView tv = new TextView(this);
    tv.setText(name);

    LinearLayout ll = (LinearLayout) this.findViewById(R.id.linear_main);
    ll.addView(tv);
  }

}
```

The xml layout file:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android (http://schemas.android.com/apk/res/android) "
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical"
        android:id="@+id/linear_main" >

        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Enter a name:" />

        <EditText
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:id="@+id/txt_name" />


        <Button
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:id="@+id/btn_confirm"
          android:text="Add Name"
          android:onclick="addName" />

    </LinearLayout>
```

Page 1 of 1

Related Topicsbeta

Activities, Intents And Something On Events

Android, Part IV: Databases And Menus

Android, Part I: Welcome To Android

Android, Part II: A Simple UI With Events

Adding A Bug Reporter / Suggestion

Box To Your Android App

Working With Resourses

Android, Part V: Signing And Publishing

GPU Gems Is Now Available Online For Free - Good Resource For Graphics Programmers

Google Demonstrated Its Android Mobile Phone Software Google I/O Confe - Screenshots: Google's Android Comes To Life

Dialogs And The Android Lifecycle