

LFE Medieninformatik • Prof. Dr. Heinrich Hußmann (Dozent), Alexander De Luca, Gregor Broll (supervisors)

Praktikum Entwicklung von Mediensystemen mit Android

Implementing a User Interface



Outline

- Introduction
- Layouting Components using XML
- Common Layout Objects
- Hooking into a Screen Element
- Listening for UI Notifications
- Applying a Theme to Your Application



Introduction

- Activity
 - Basic functional unit of an Android application
 - But by itself, it does not have any presence on the screen
- Views and Viewgroups
 - Basic units of user interface expression on the Android platform

Views

Implementing a User Interface



LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Views

Introduction

- android.view.View
 - Stores layout and content for a specific rectangular area of the screen
 - Handles measuring and layout, drawing, focus change, scrolling, and key/gestures
 - Base class for widgets
 - Text
 - EditText
 - InputMethod
 - MovementMethod
 - Button
 - RadioButton
 - Checkbox
 - ScrollView

```
package com.android.hello;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

Viewgroups

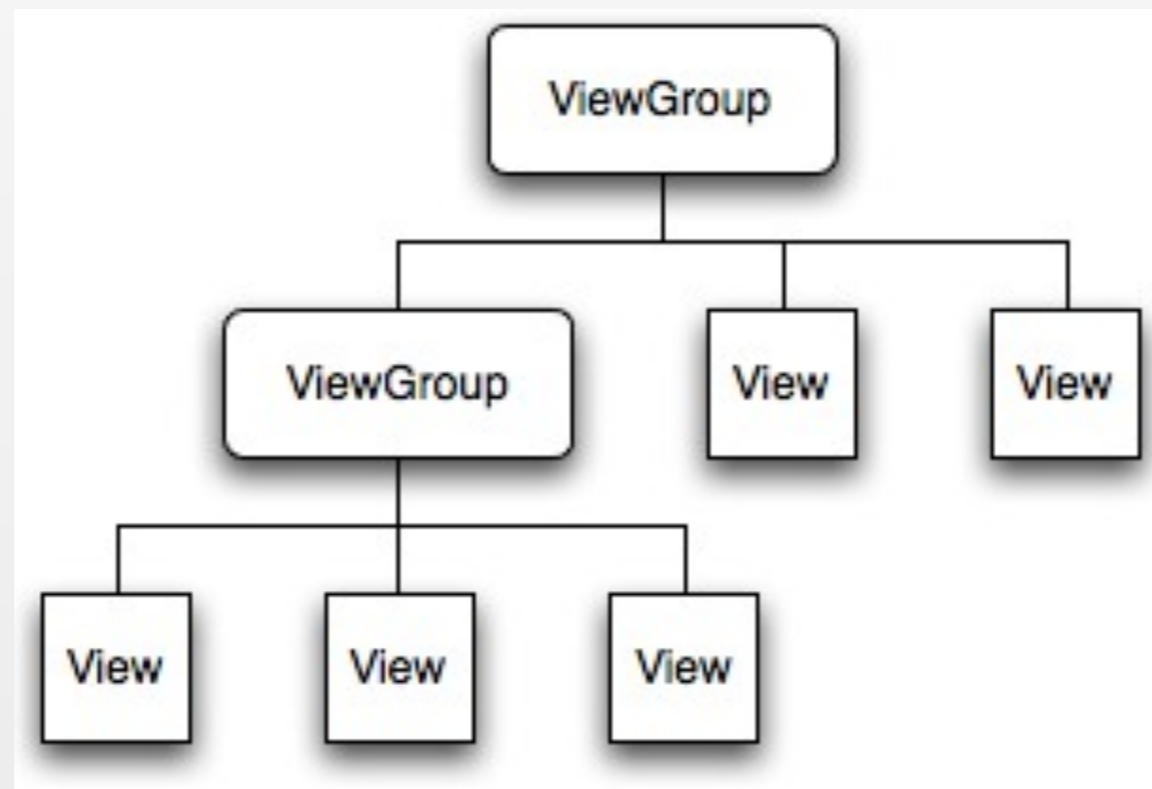
Introduction

- `android.view.ViewGroup`
 - Contains and manages a subordinate set of views and other viewgroups
 - Base class for layouts

Tree-Structured UI

Introduction

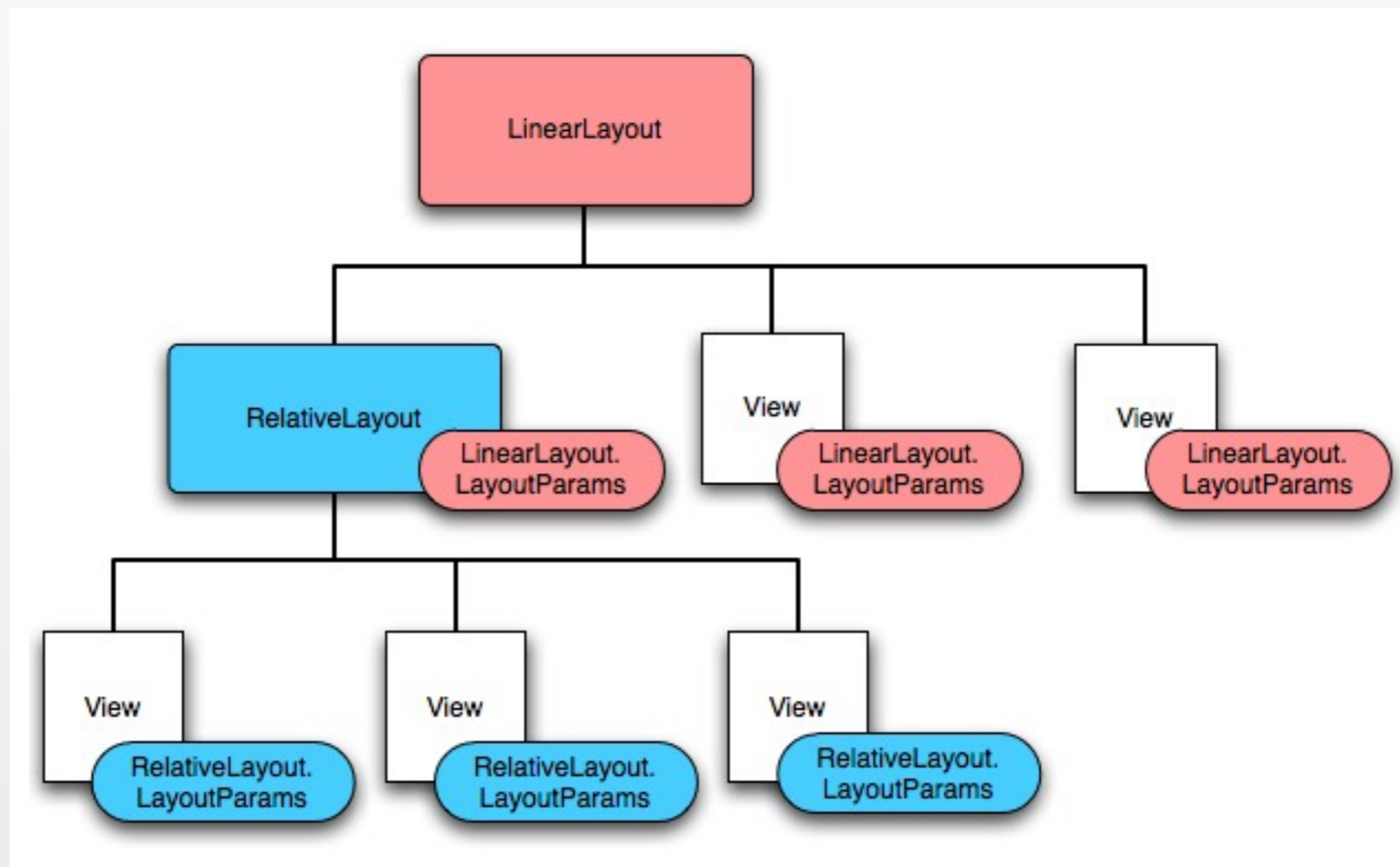
- An Activity in Android
 - Defined using a tree of view and viewgroup nodes
- setContentView() method
 - Called by the Activity to attach the tree to the screen for rendering



LayoutParams

Introduction

- Every viewgroup class uses a nested class that extends `ViewGroup.LayoutParams`
 - Contains property types that defines the child's size and position



Programmatic UI Layout

- Programmatic UI Layout

- Constructing and building the applications UI directly from source code
- Disadvantage
 - small changes in layout can have a big effect on the source code

```
package com.android.hello;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText("Hello, Android");
        setContentView(tv);
    }
}
```

Upgrading UI to XML Layout

- XML-based Layout

- Inspired by web development model where the presentation of the application's UI is separated from the logic
- Two files to edit
 - Java file – application logic
 - XML file – user interface

Upgrading UI to XML Layout

The screenshot displays the Eclipse IDE interface for an Android project named 'HelloAndroidProject'. The Package Explorer on the left shows the project structure, with 'UIExample.java' and 'main.xml' circled in red. The main editor shows the Java code for 'UIExample.java', with 'super.onCreate(savedInstanceState);' and 'setContentView(R.layout.main);' circled in red. The Outline view on the right shows the class structure. The bottom console shows a DDMS log message: '[2008-03-18 14:14:20 - adb] * daemon not running. starting it now *'.

```
package pem.samplecode.ui;

import android.app.Activity;

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

Upgrading UI to XML Layout

The image shows two overlapping windows from an IDE, illustrating the process of upgrading a UI from a visual design to XML code.

The top window shows the 'Design' view of a layout file named 'main.xml'. The layout is a `LinearLayout` containing a `TextView`. The `LinearLayout` has the following attributes:

- `xmlns:android`: `http://schemas.android.com/apk/res/android`
- `android:orientation`: `vertical`
- `android:layout_width`: `fill_parent`
- `android:layout_height`: `fill_parent`

The `TextView` has the following attributes:

- `android:layout_width`: `fill_parent`
- `android:layout_height`: `wrap_content`
- `android:text`: `"Hello World, UIExample"`

The bottom window shows the 'Source' view of the same 'main.xml' file, displaying the corresponding XML code:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello World, UIExample"
    />
</LinearLayout>
```



Common Layout Objects

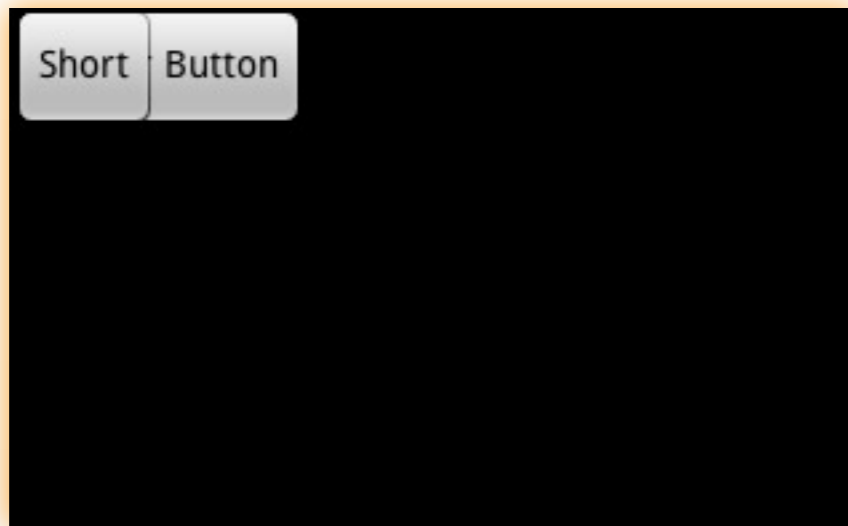
Implementing a User Interface



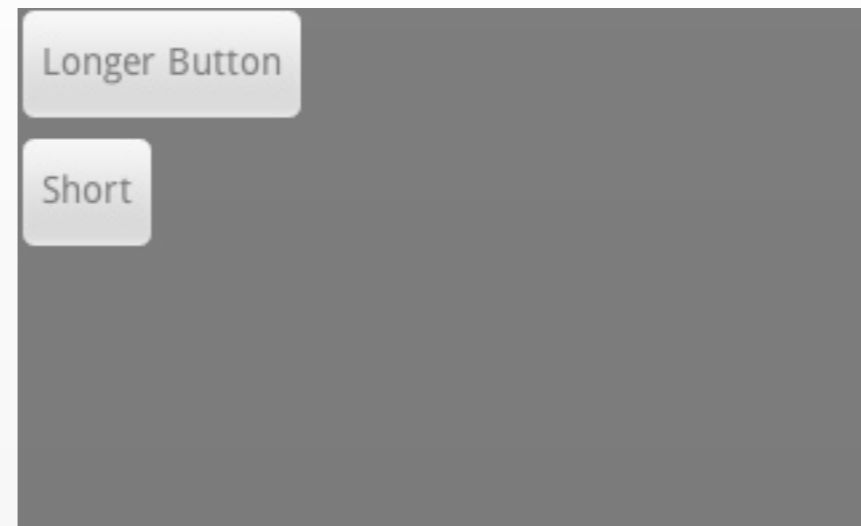
LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

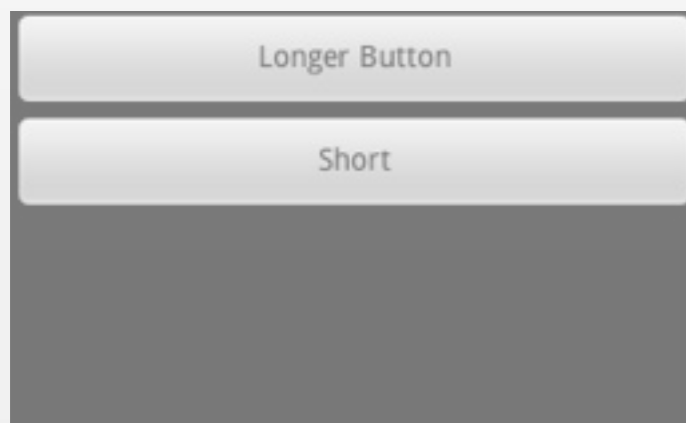
Common Layout Objects



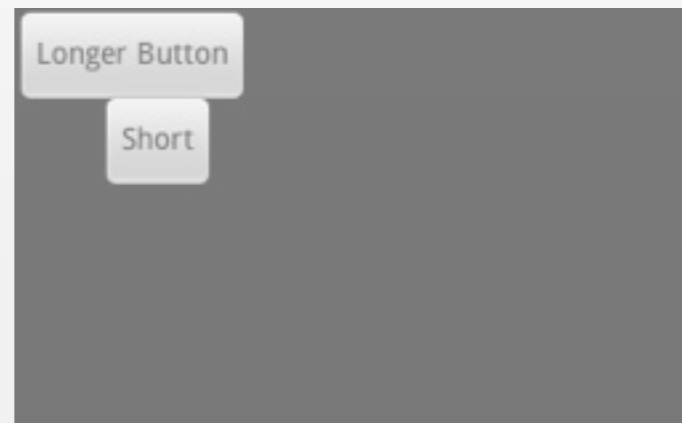
FrameLayout



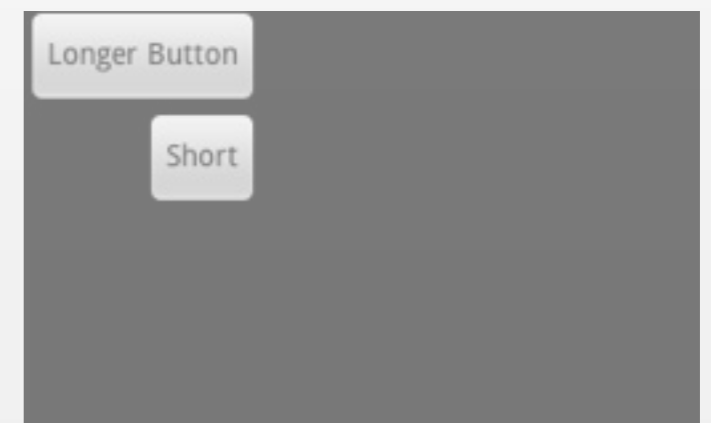
LinearLayout



TableLayout



AbsoluteLayout



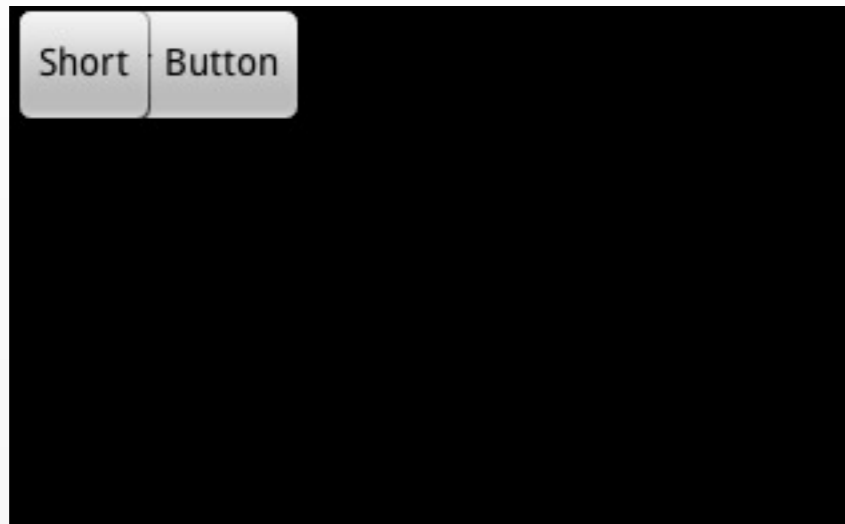
RelativeLayout

FrameLayout

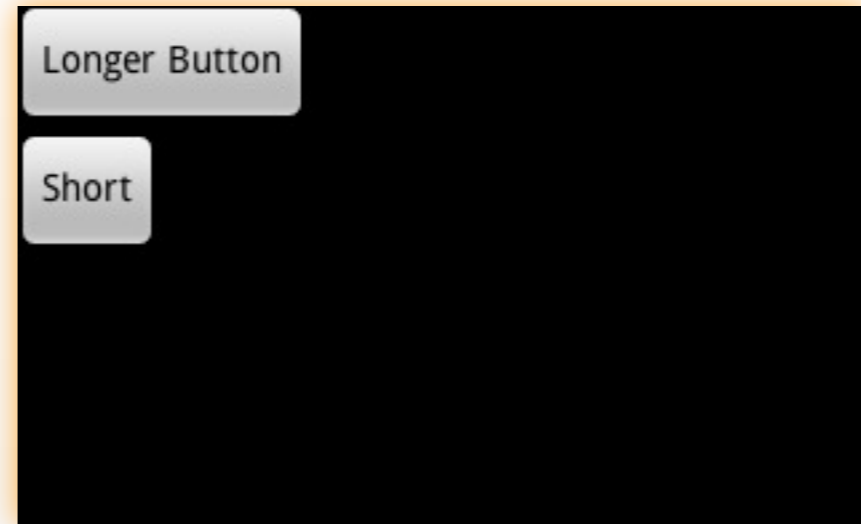
Common Layout Objects

- Simplest layout object
- Intended as a blank reserved space on your screen that you can later fill with a single object
 - Example: a picture that you'll swap out
- All child elements are pinned to the top left corner of the screen
- Cannot specify a location for a child element

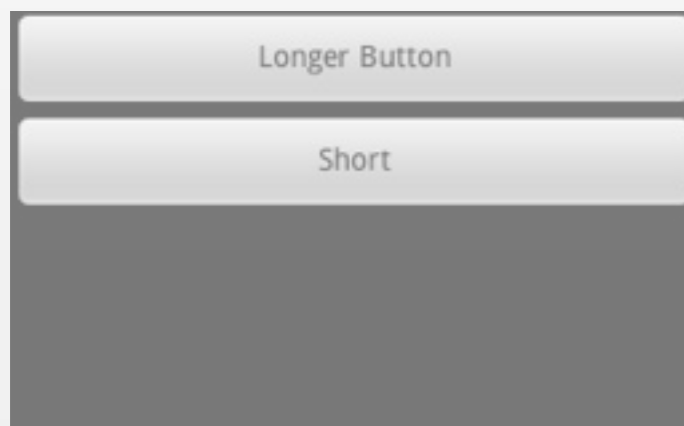
Common Layout Objects



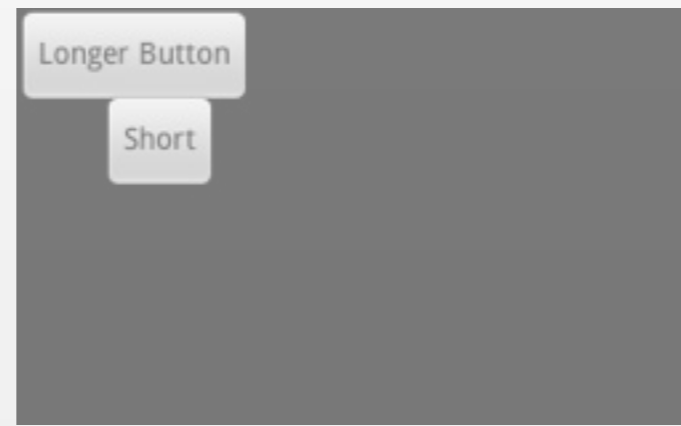
FrameLayout



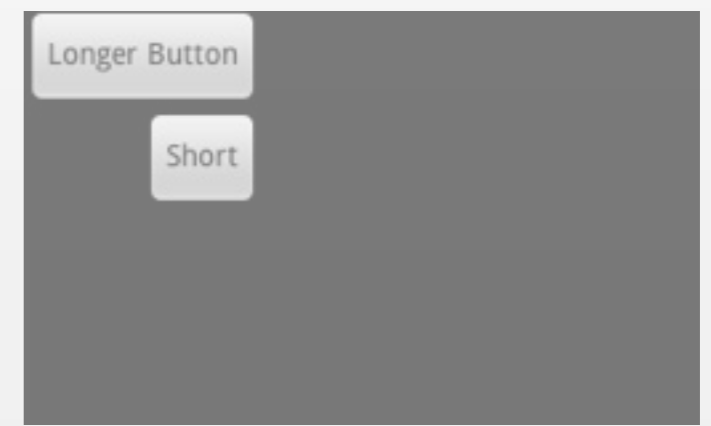
LinearLayout



TableLayout



AbsoluteLayout

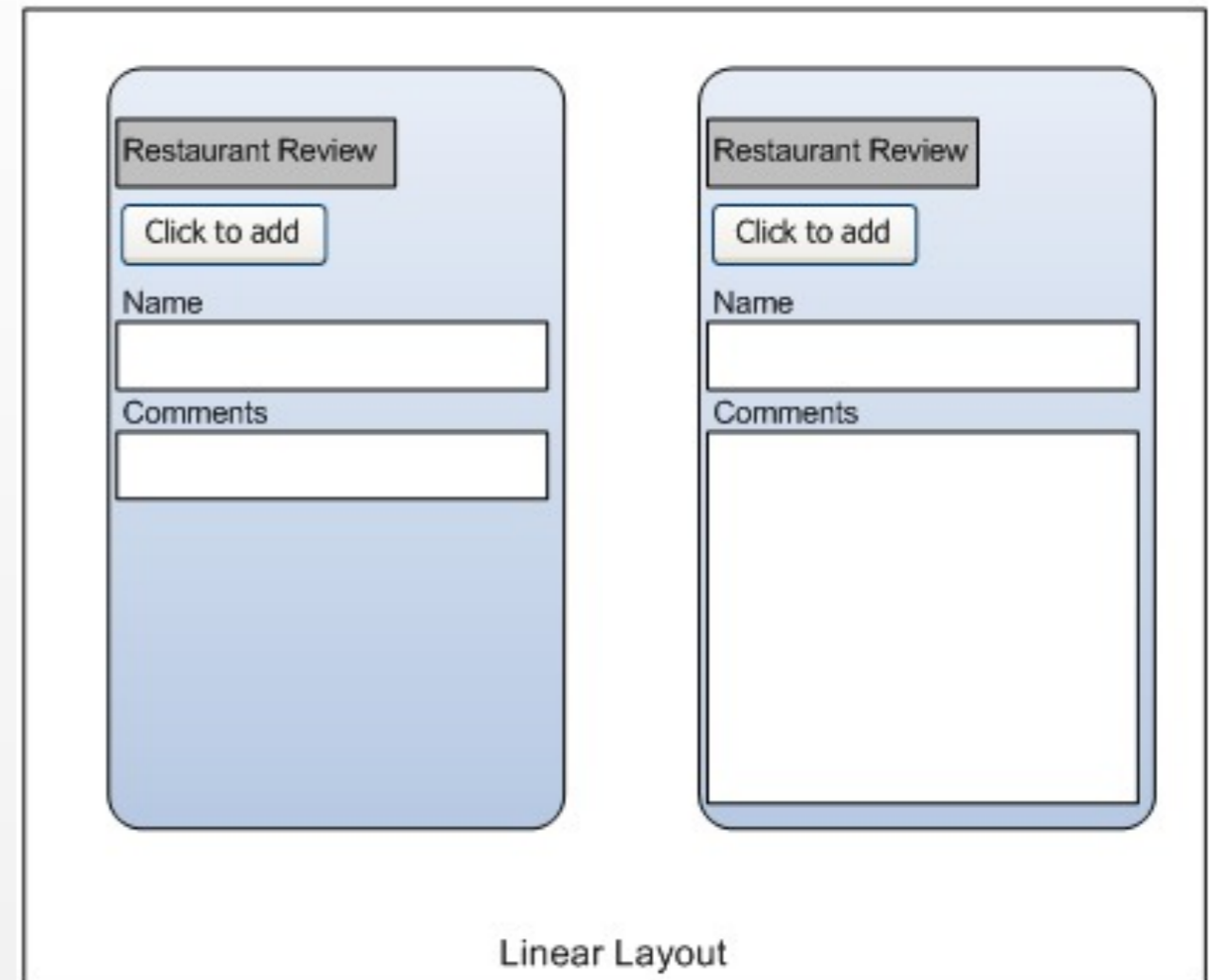


RelativeLayout

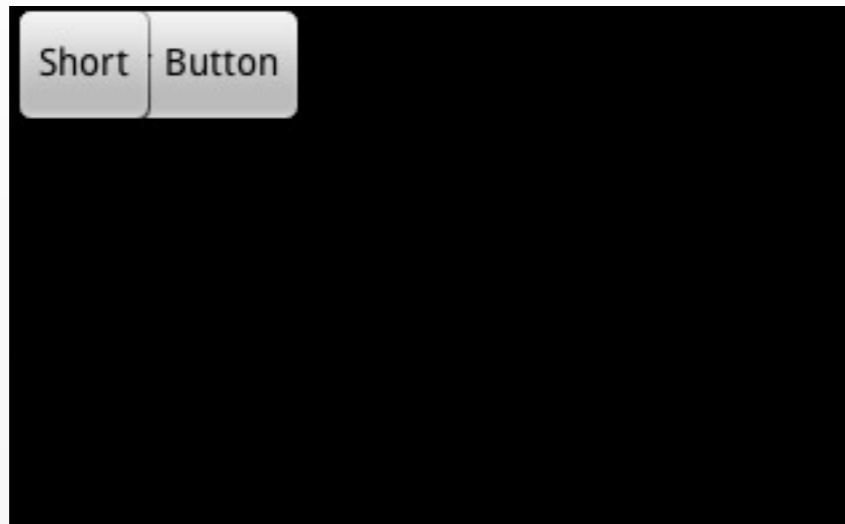
LinearLayout

Common Layout Objects

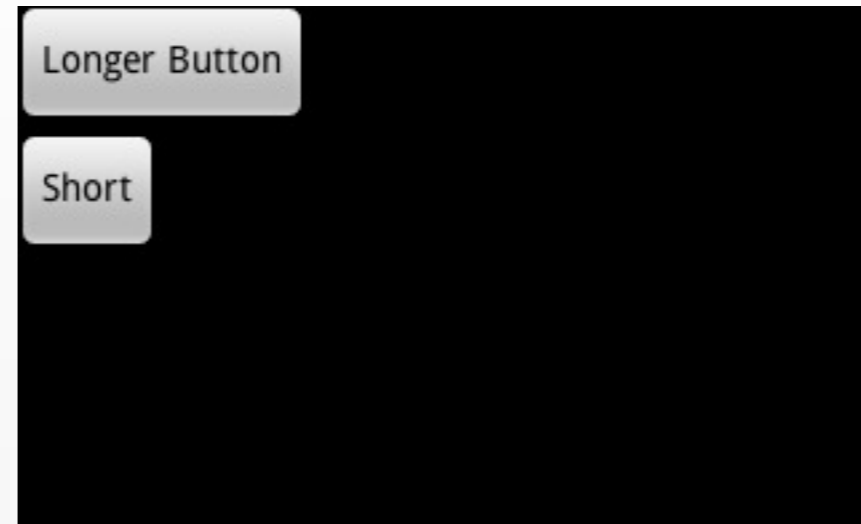
- Aligns all children in a single direction — vertically or horizontally
 - All children are stacked one after the other
 - a vertical list will only have one child per row (no matter how wide they are)
 - a horizontal list will only be one row high (the height of the tallest child, plus padding)



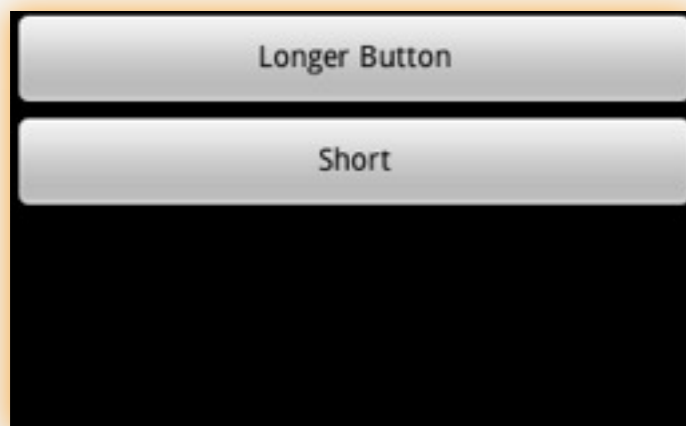
Common Layout Objects



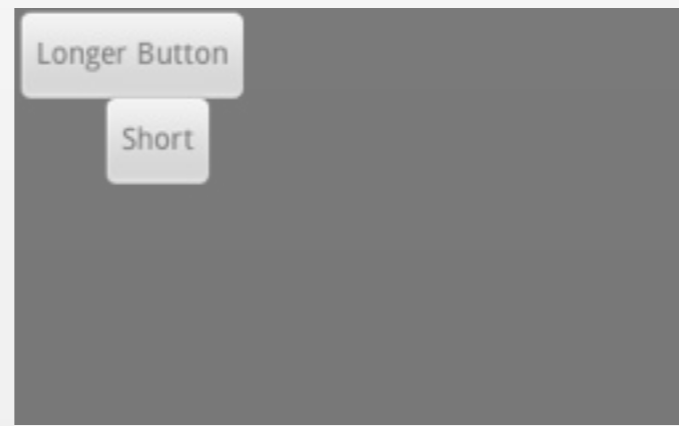
FrameLayout



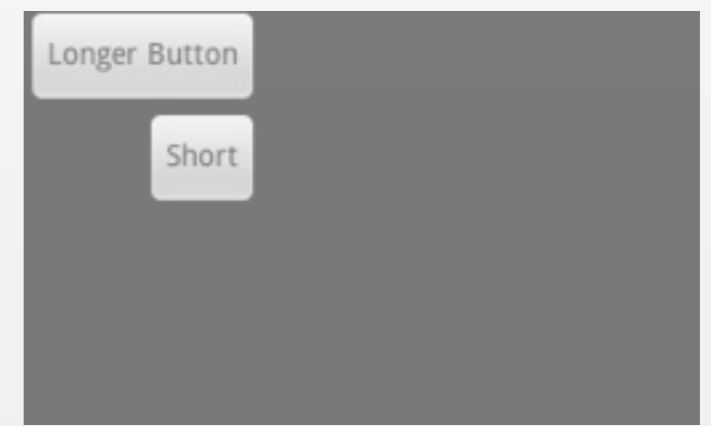
LinearLayout



TableLayout



AbsoluteLayout

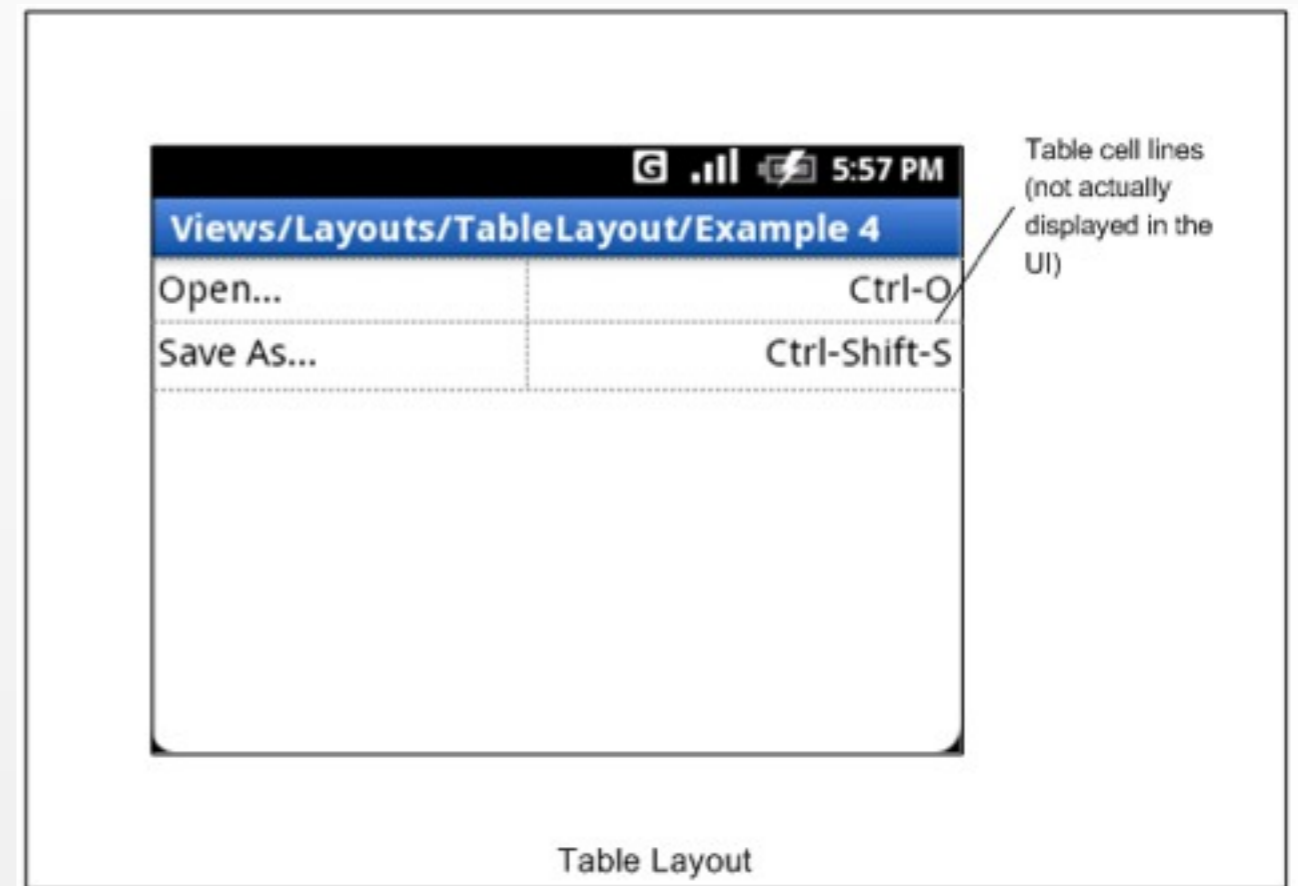


RelativeLayout

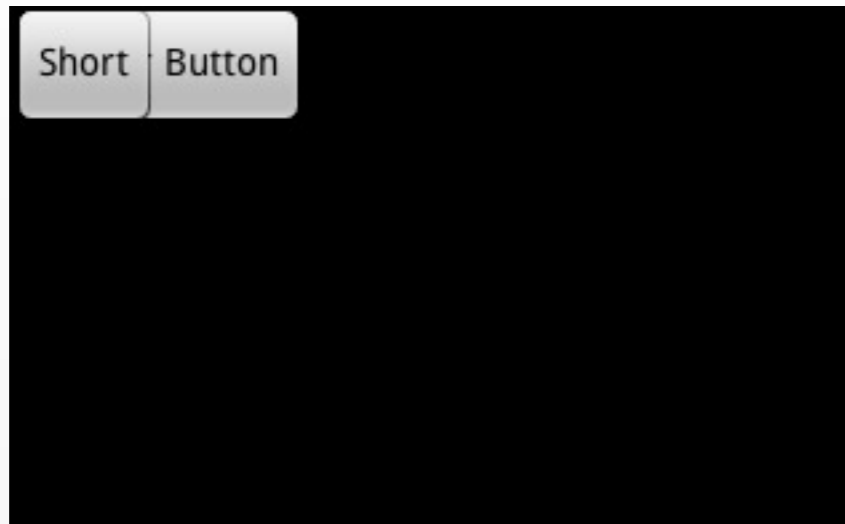
TableLayout

Common Layout Objects

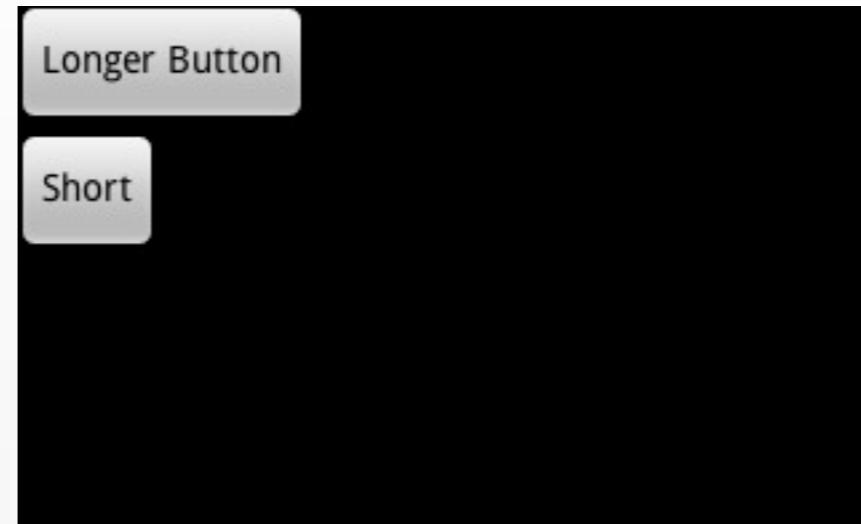
- Positions its children into rows and columns
- Does not display border lines for their rows, columns, or cells
- Cells cannot span columns, as they can in HTML



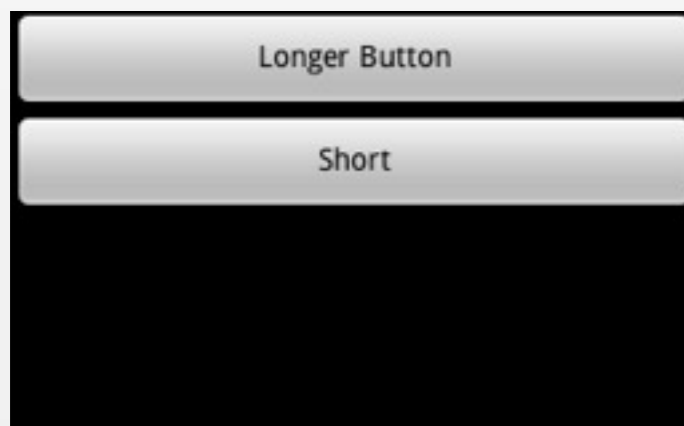
Common Layout Objects



FrameLayout



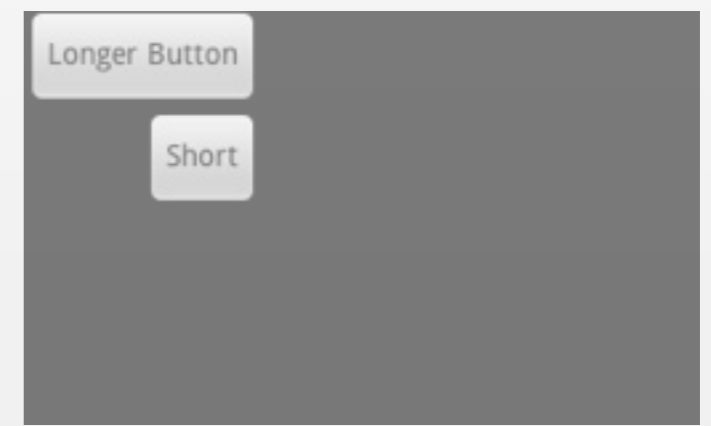
LinearLayout



TableLayout



AbsoluteLayout



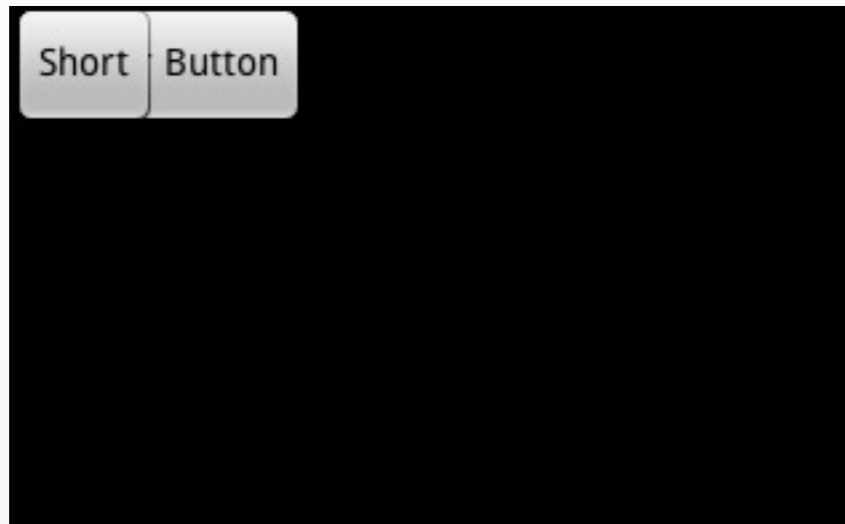
RelativeLayout

AbsoluteLayout

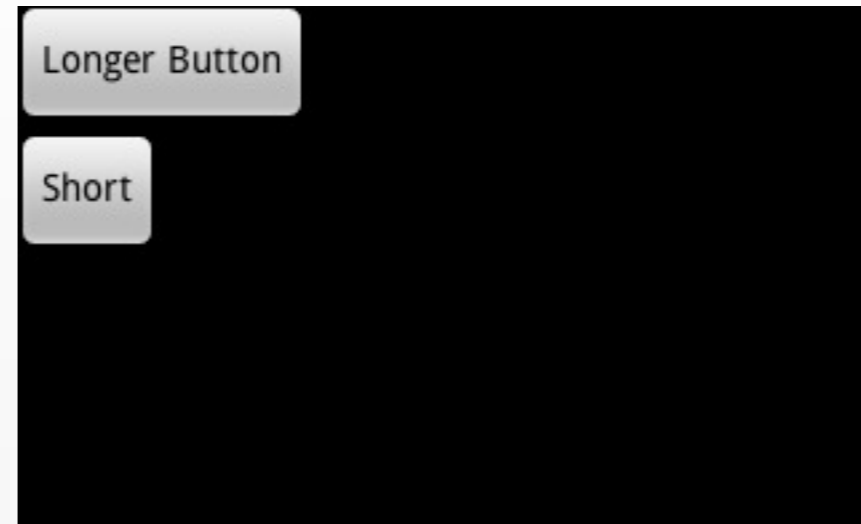
Common Layout Objects

- Enables children to specify exact x/y coordinates to display on the screen
 - (0,0) is the upper left corner
 - values increase as you move down or to the right
- Overlapping elements are allowed (although not recommended)
- NOTE:
 - It is generally recommended NOT to use AbsoluteLayout UNLESS you have good reasons to use it
 - It is because it is fairly rigid and does not work well with different device displays

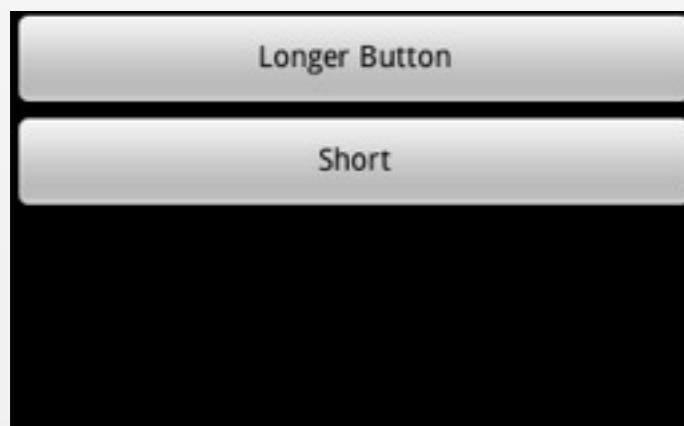
Common Layout Objects



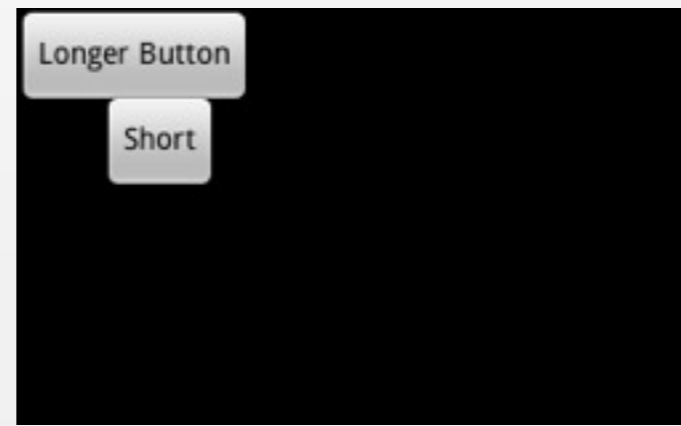
FrameLayout



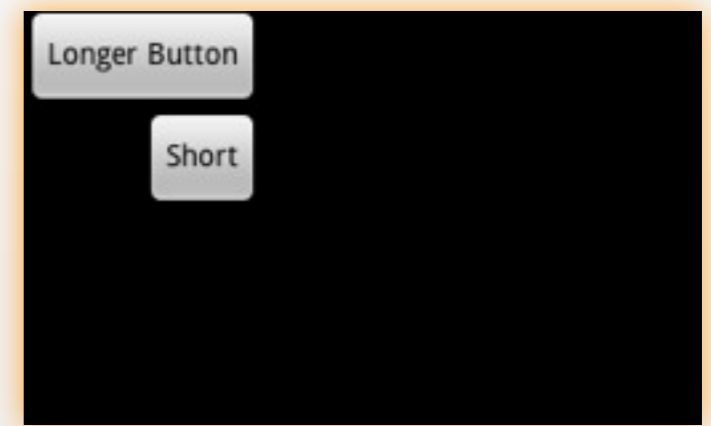
LinearLayout



TableLayout



AbsoluteLayout

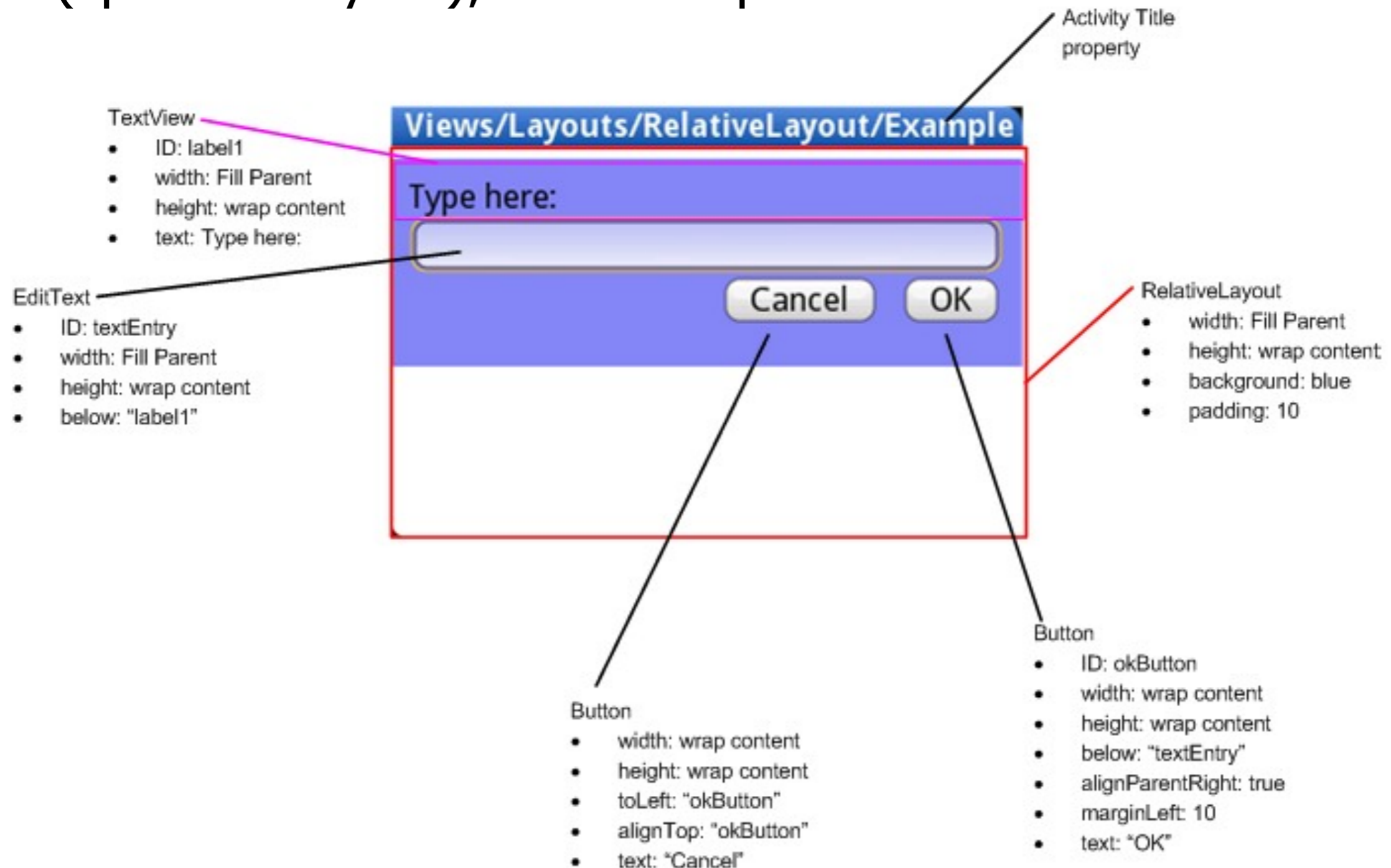


RelativeLayout

RelativeLayout

Common Layout Objects

- Lets children specify their position relative to each other (specified by ID), or to the parent



Important Layout Paramters

Allgemein:

Layout-Height:	fill_parent, wrap_content, Pixels
Layout-Width:	fill_parent, wrap_content, Pixels
Id:	@+id/my_variable
Min-Height, Max-Height...	
Min-Width, Max-Width	

Speziell:

EditText	Input type	text, textEmailAddress, number, numberDecimal
TextView, Button, EditText	Text	@string/resource_id
TextView	Text color, Text size	

Online Reference

<http://developer.android.com/guide/tutorials/views/index.html>

The screenshot shows the Android Developer website's 'Hello, Views' tutorial page. The page is titled 'Hello, Views' and contains an introduction to the concept of Views in Android. It lists several types of Views: **LinearLayout**, **RelativeLayout**, **TextView**, **DatePicker**, **TimePicker**, **EditText**, **Spinner**, **AdapterView**, **ListView**, **GridView**, **Gallery**, **TabHost**, **MapView**, and **WebView**. Each view type is represented by a small image showing its appearance on a mobile device. The page also includes a navigation menu on the left and a footer with the date 'Donnerstag, 6. Mai 2010'.

Hooking into a screen element

Implementing a User Interface



LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Hooking into a Screen Element



Text field

Button

Hooking into a Screen Element



```
UIExample.java  main.xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello World, UIExample"
    />

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

    <Button
        android:id="@+id/ok"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="OK"
    />

</LinearLayout>
```

Design Source

Hooking into a Screen Element

```
<E-----  
  android:id="@+id/name_entry"  
  an-----  
  android:layout_height="wrap_content"  
 />  
  
<Button  
  android:id="@+id/ok"  
  android:layout_width="fill_parent"  
  android:layout_height="wrap_content"  
  android:text="OK"  
 />
```

Any String value
(no spaces)

@+id syntax:

Creates a resource number in the R class (R.java file) if one doesn't exist, or uses it if it does exist.

Hooking into a Screen Element

```
UIExample.java  main.xml  main2.xml  R.java X
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package pem.samplecode.ui;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int name_entry=0x7f050000;
        public static final int ok=0x7f050001;
    }
    public static final class layout {
        public static final int main=0x7f030000;
        public static final int main2=0x7f030001;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
    }
}
}
```

Hooking into a Screen Element

```
UIExample.java x main.xml
package pem.samplecode.ui;

import android.app.Activity;
import android.os.Bundle;
import android.widget.EditText;

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

        //Add a handle to UI components
        EditText nameEntry = (EditText) findViewById(R.id.name_entry);
        nameEntry.setText("Enter your name here");
    }
}
```


Hooking into a Screen Element



Listening for UI Notifications

```
UIExample.java x main.xml
package pem.samplecode.ui;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

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

        //Add a handle to UI components
        EditText nameEntry = (EditText)findViewById(R.id.name_entry);
        nameEntry.setText("Enter your name here");

        Button okButton = (Button)findViewById(R.id.ok);

        //Create an anonymous class to act as a button click listener
        okButton.setOnClickListener(new OnClickListener()
        {
            public void onClick(View v) {
                setResult(RESULT_OK, "Done!");
                finish();
            }
        });
    }
}
```

Resource Folders and Localization

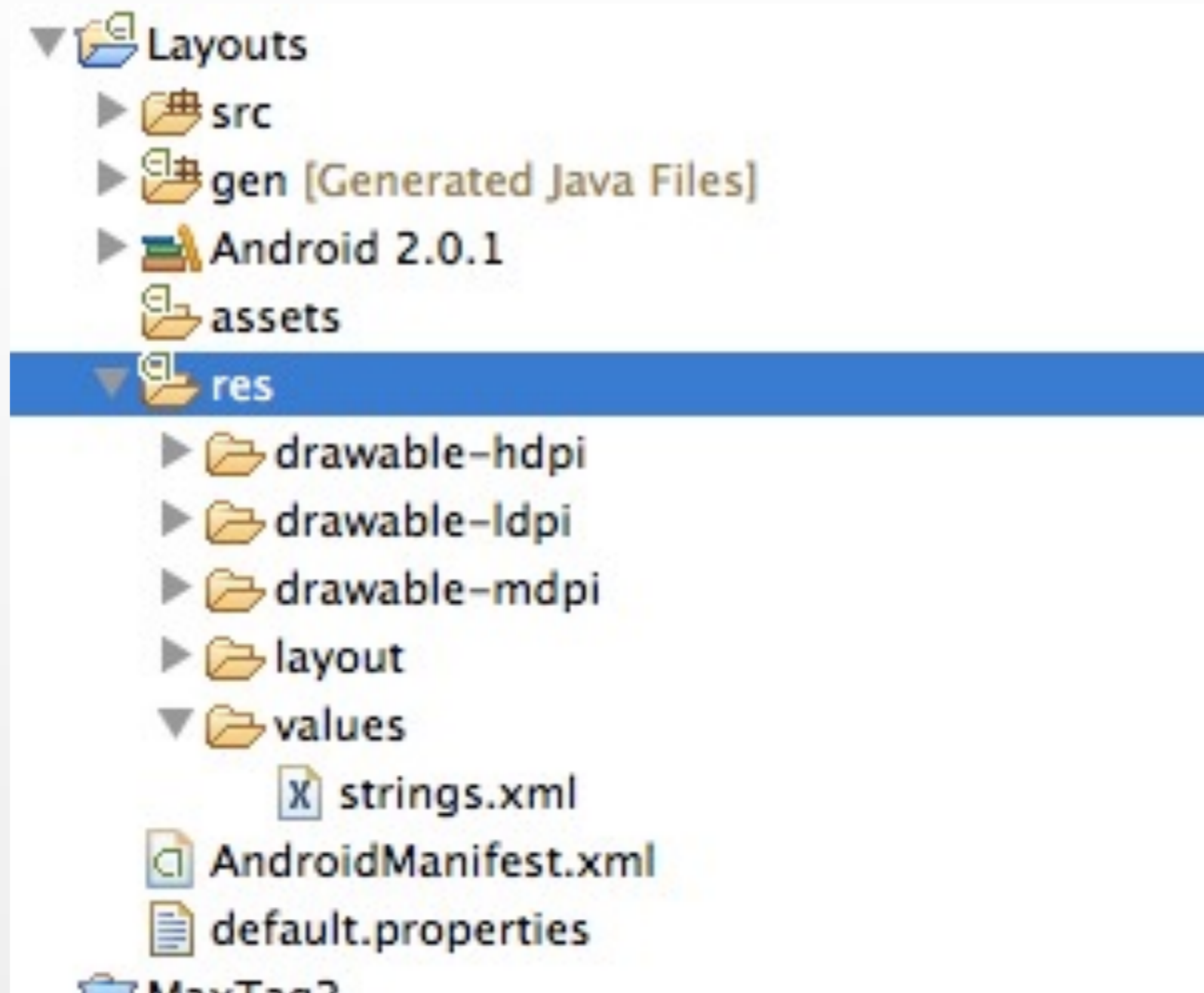
Implementing a User Interface



LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Resource Folders



Resource Folders

- Folder structure is automatically parsed into Resource-File
- Do not modify this file!



```
⊕/* AUTO-GENERATED FILE. DO NOT MODIFY.⋮  
package de.maxmaurer.layouts;  
  
public final class R {  
    ⊖ public static final class attr {  
        }  
    ⊖ public static final class drawable {  
        public static final int icon=0x7f020000;  
    }  
    ⊖ public static final class id {  
        public static final int Button01=0x7f050001;  
        public static final int Button02=0x7f050000;  
    }  
    ⊖ public static final class layout {  
        public static final int main=0x7f030000;  
    }  
    ⊖ public static final class string {  
        public static final int app_name=0x7f040001;  
        public static final int hello=0x7f040000;  
    }  
}
```

Resource Folders

- Separate storage of Strings and Graphics
- Makes it easier to modify software parts
- Resources are accessed via „R.java“

```
package de.maxmaurer.layouts;

+import android.app.Activity;

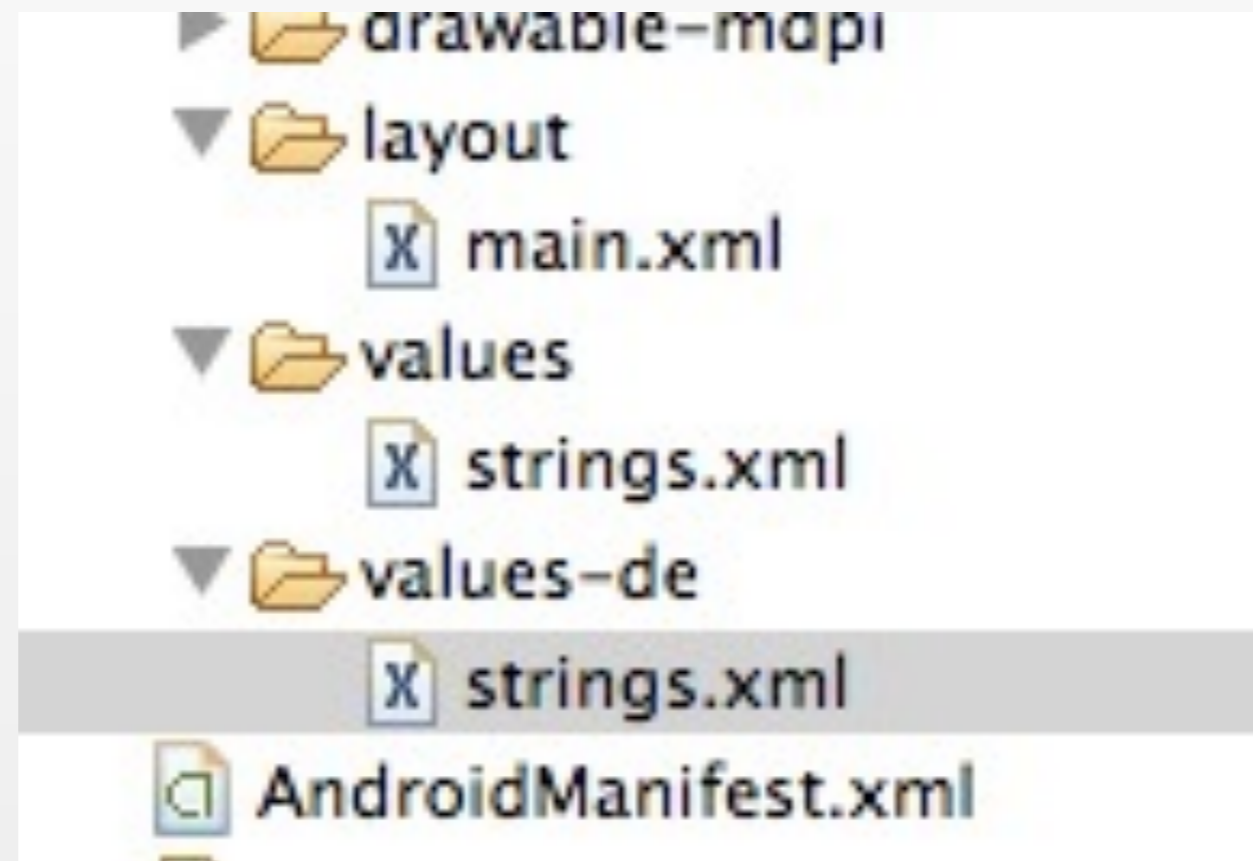
public class Layouts extends Activity {
    /** Called when the activity is first created. */
    @Override
    -public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        View v = findViewById(R.layout.main);
        TextView tv = new TextView(this);
        tv.setText(getString(R.string.hello_world));
        setContentView(v);
    }
}
```

Resource Folders

The screenshot displays the 'Android Resources (default)' window in an IDE. The 'Resources Elements' tab is active, showing a list of string resources: 'hello (String)', 'app_name (String)', and 'hello_world (String)'. The 'hello_world (String)' resource is selected. To the right of the list are buttons for 'Add...', 'Remove...', 'Up', and 'Down'. Below these buttons, a text field contains 'hello_world' and another contains 'Hello, World!'. The 'Layouts' tab is also visible, showing a preview of a layout with the text 'Hello, World!'. The system tray at the top right shows a 3G signal icon, a battery icon, and the time '1:56 PM'.

Localization

- Creating folders for other languages does not need any code change
- Watch the application size!



Localization

Android Resources (de)

Resources Elements: (S) (C) (D) (D) (S) (I) (S) (I) Az

hello_world (String)

Add...
Remove...
Up
Down

Attributes for hello_world (String)

Strings, with optional simple formatting, are used as resources. You can add formatting using three standard HTML tags: **b**, *i*, or ``. If you use a quote in your string, you must escape the whole string in the other kind.

Name* hello_world
Value* Hallo, Welt!

3G [signal] [battery] 13:55

Layouts

Hallo, Welt!

Localization

- May be used for other device specific things as well
 - Country
 - Screen dimensions
 - Screen orientation
 - Touchscreen type (finger, stylus)
 - and many more

```
MyApp/  
  res/  
    drawable-en-rUS-large-long-port-mdpi-finger-keysexposed-qwerty-navexposed-dpad-480x320/
```

Applying a Theme to your application

Implementing a User Interface



LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Applying a Theme to Your Application

- Default theme: `android.R.style.Theme`
 - <http://developer.android.com/reference/android/R.style.html>
- Two ways to set the theme
 - Adding the theme attribute in `AndroidManifest.xml`
 - Calling `setTheme()` inside the `onCreate()` method

Editing AndroidManifest.xml

Applying a Theme to Your Application

- Adding the theme attribute in AndroidManifest.xml



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="pem.samplecode.ui">
    <application android:icon="@drawable/icon"
        android:theme="@android:style/Theme.Black">
        <activity android:name=".UIExample" 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>
```

Calling setTheme()

Applying a Theme to Your Application

- Calling setTheme() inside the onCreate() method



```
UIExample.java x main.xml x AndroidManifest.xml
package pem.samplecode.ui;

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

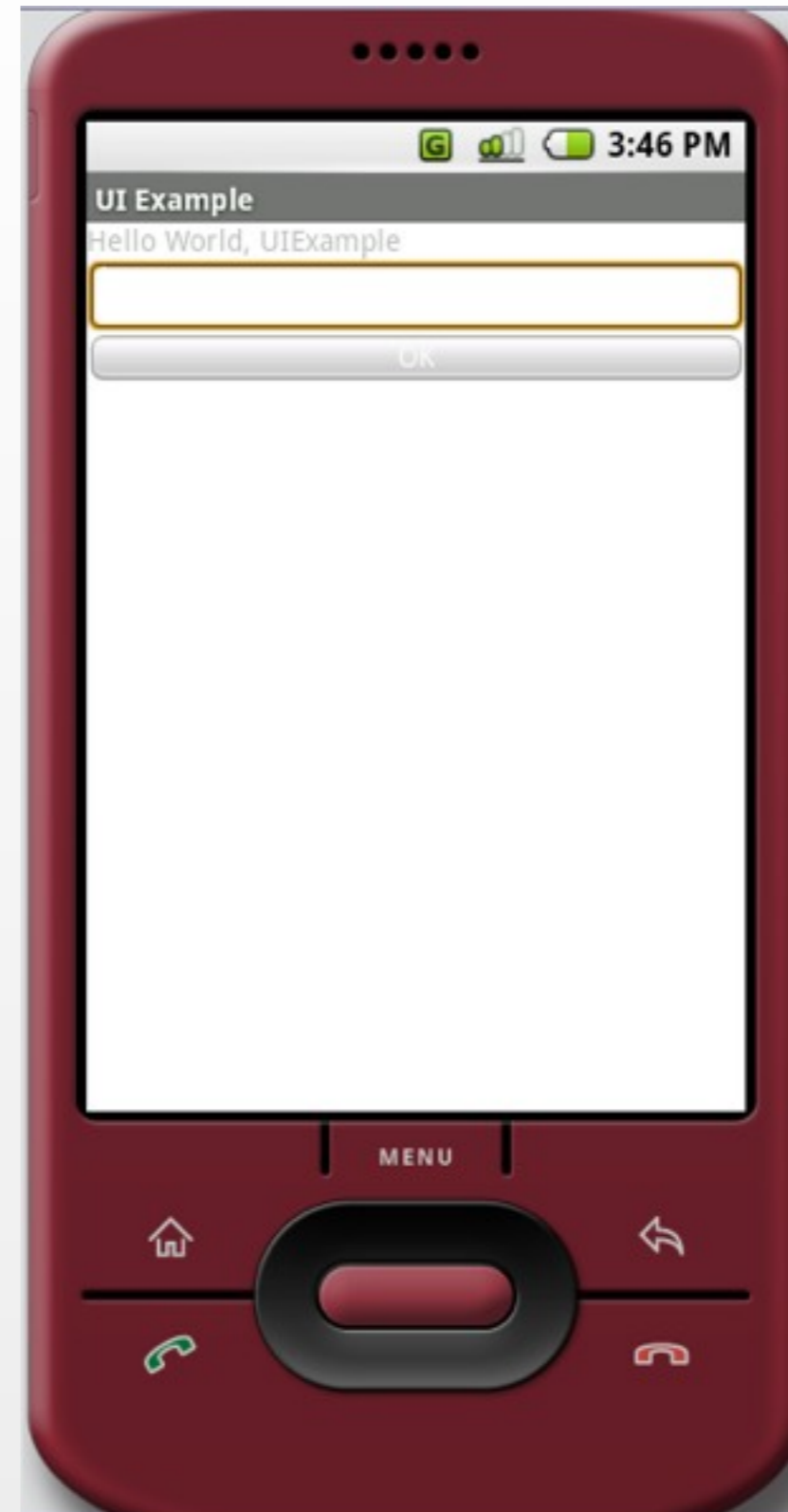
public class UIExample extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setTheme(android.R.style.Theme_Black);
        setContentView(R.layout.main);
    }
}
```

Theme_Black



Theme_Light



Exercise

Implementing a User Interface



LMU

LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

Exercise

- Chatting with Myself Application

- Components

- Image
- Text that displays the Name
- Drop down box that displays the status
 - Available
 - Busy
 - Away
- Text Area that displays messages
 - Format for the output is
 - ChatterName:The Message
- Text Field where the user can type the message
- Send Button
 - When Pressed
 - Message typed on the text field is displayed on the text area
 - Text field is cleared

- Any improvements on the design or additional functionality is encouraged



**Fragen?
Viel Spaß!**

