CS 646 Android Mobile Application Development
Spring Semester, 2015
Doc 4 More Android Basics
Jan 29, 2014

How to connect UI widgets to code

Two Directions

How does code get reference to UI elements

How do UI elements call code

How does code get reference to UI elements

If create UI elements in code - have reference

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    LinearLayout layout = new LinearLayout(this);
    Button test = new Button(this);
    test.setText("Hello");
    layout.addView(test,
        new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT,
            0));
    setContentView(layout);
}
```

How does code get reference to UI elements

If create UI in XML layout - id

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Button test = (Button) this.findViewById(R.id.test);
}
```

```
<Button
    android:id="@+id/test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

How do UI elements call code

In xml layout - onXXX

```
<Button
    android:id="@+id/test"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    android:onClick="makeToast"/>
```

```
public class CreateUIInCodeActivity extends Activity {

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

    public void makeToast(View source) {
      Toast.makeText(this, "Hello World", Toast.LENGTH_SHORT).show();
    }
}
```

How do UI elements call code

In code - Listeners

GestureDetector.OnGestureListener	Notify when gestures occur
MenuItem.OnMenuItemClickListener	a menu item is clicked.
View.OnClickListener	a view is clicked.
View.OnCreateContextMenuListener	the context menu for this view is being built.
View.OnFocusChangeListener	the focus state of a view changed.
View.OnKeyListener	a key event is dispatched to this view.
View.OnLongClickListener	a view has been clicked and held.
View.OnTouchListener	a touch event is dispatched to this view.
ViewGroup.OnHierarchyChangeListener	the hierarchy within this view changed.
ViewStub.OnInflateListener	ViewStub has successfully inflated its layout resource.
ViewTreeObserver.OnGlobalFocusChangeListener	the focus state within the view tree changes.
ViewTreeObserver.OnGlobalLayoutListener	the global layout state or the visibility of views within the view tree changes.
ViewTreeObserver.OnPreDrawListener	the view tree is about to be drawn.
ViewTreeObserver.OnTouchModeChangeListener	the touch mode changes.

See View overview at: http://code.google.com/android/reference/android/view/package-summary.html for more information and links to each listener

What is a Listener?

Java Interface

View.OnClickListener

abstract void onClick(View v)
Called when a view has been clicked.

UI elements call methods on concrete Listener object in response to user action.

Have to add the concrete Listener to UI element

Using the Listener

```
public class CreateUIInCodeActivity extends Activity implements View.OnClickListener{
      Button test;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        test = (Button) this.findViewById(R.id.test);
        test.setOnClickListener(this);
    }

    public void onClick(View source) {
      Toast.makeText(this, "Hello World", Toast.LENGTH_SHORT).show();
    }
}
```

What if we have Multiple Things to click

```
public class CreateUIInCodeActivity extends Activity implements View.OnClickListener{
    Button test;

  @Override
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     this.findViewById(R.id.test);
     test.setOnClickListener(this);
      this.findViewById(R.id.OtherButton).setOnClickListener(this);
  }

  public void onClick(View source) {
    if (source == test )
          handle button test click
   else
        handle other other case
  }
}
```


It does not take too many cases for the onClick method to become ugly. The xml onclick is cleaner, but it does hide the connection in the layout file. Or one can use anonymous classes

Using an Anonymous Class

```
public class CreateUIInCodeActivity extends Activity  {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button test = (Button) this.findViewById(R.id.test);
        test.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View source) {
                        makeToast();
                    }
        });
    }

    public void makeToast() {
      Toast.makeText(this, "Hello World", Toast.LENGTH_SHORT).show();
    }
```


The anonymous class allows us to specify which method to call instead of onClick, but syntax is verbose and ugly.

Layouts

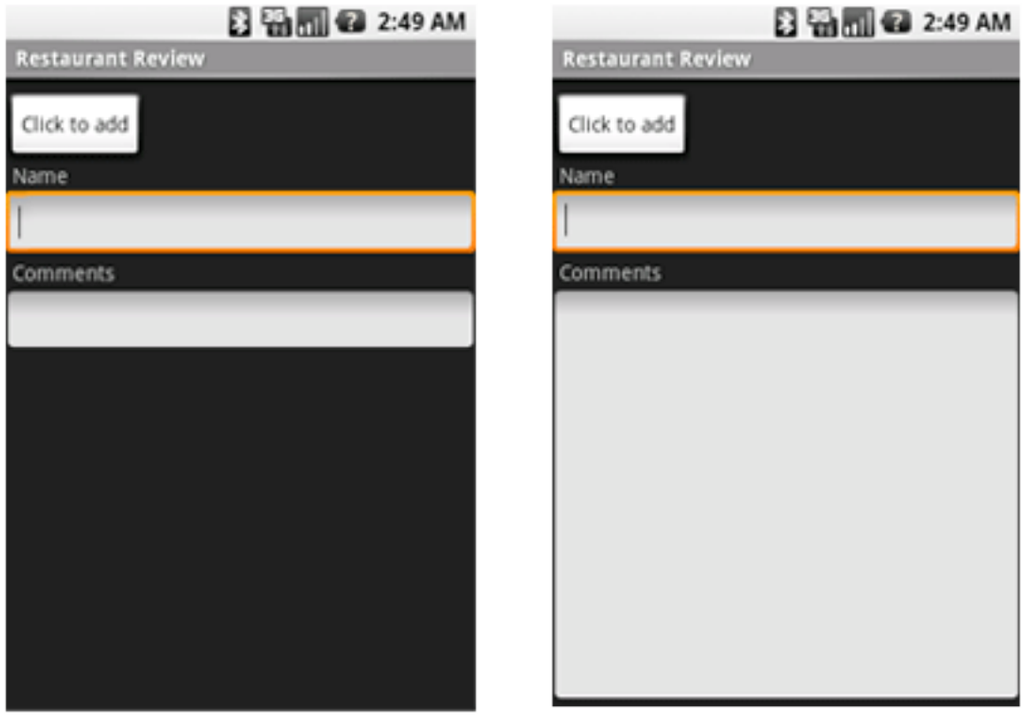
Layouts

Organize UI elements on screen

Common Layouts

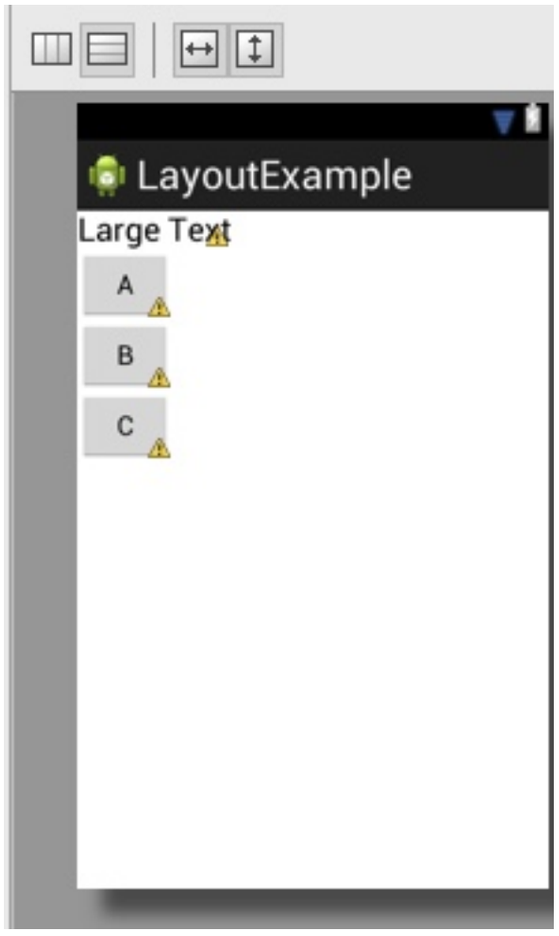
LinearLayout

Items stacked vertically or horizontally

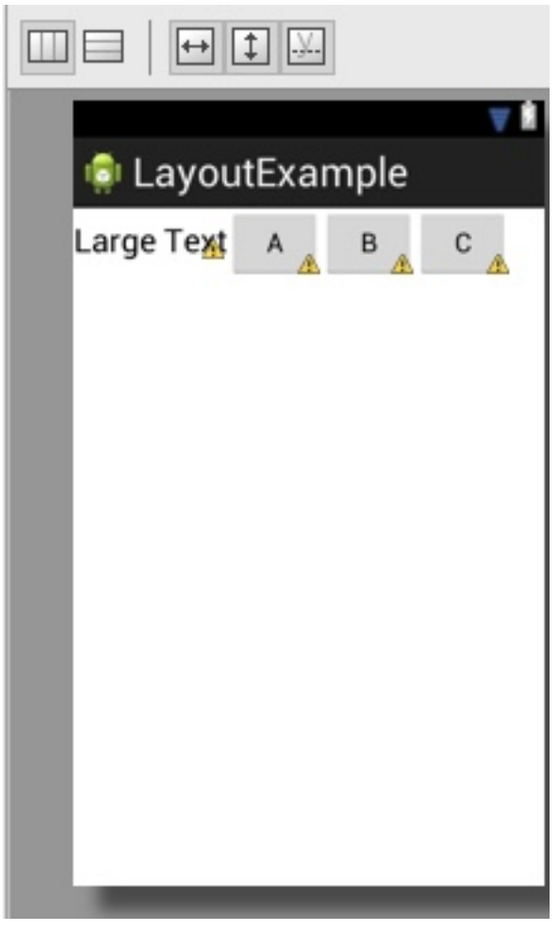


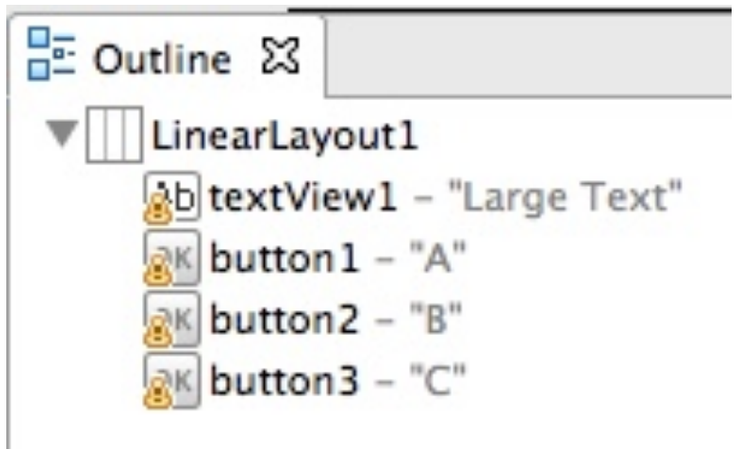
LinearLayout

vertical

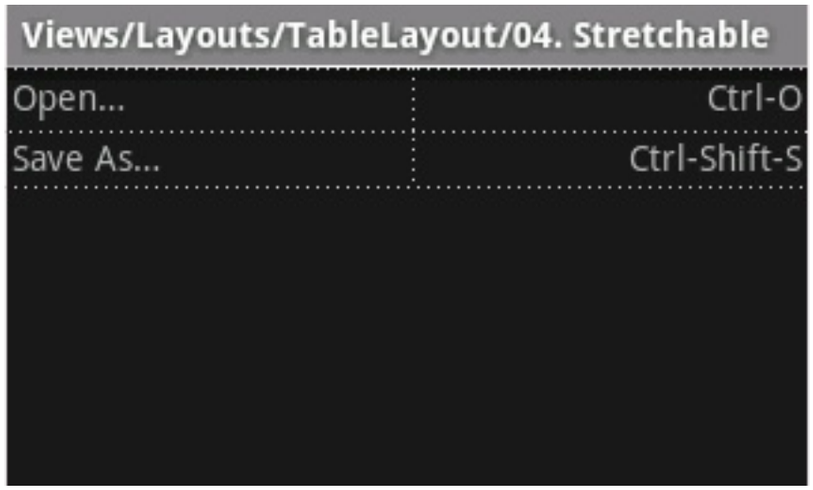


horizontal





Common Layouts



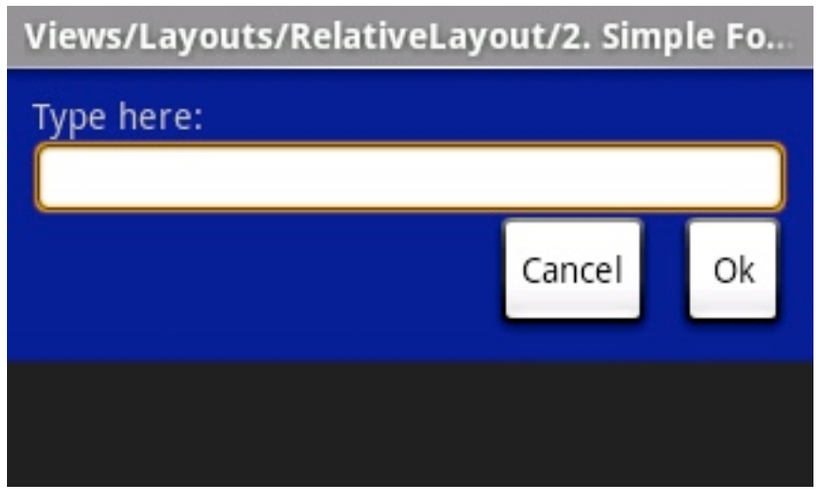
Tablelayout

In editor

- Set layout to Tablelayout
- Add TableRows to table
- Add widgets to TableRows
- Android figures out how many columns

Common Layouts

RelativeLayout

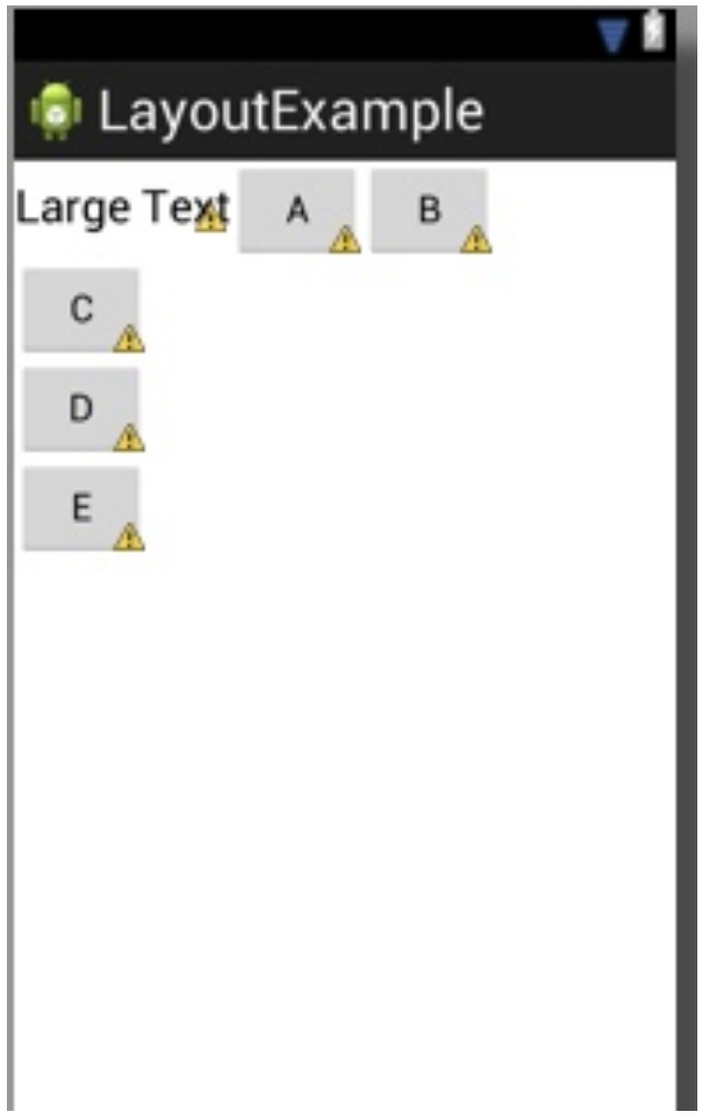


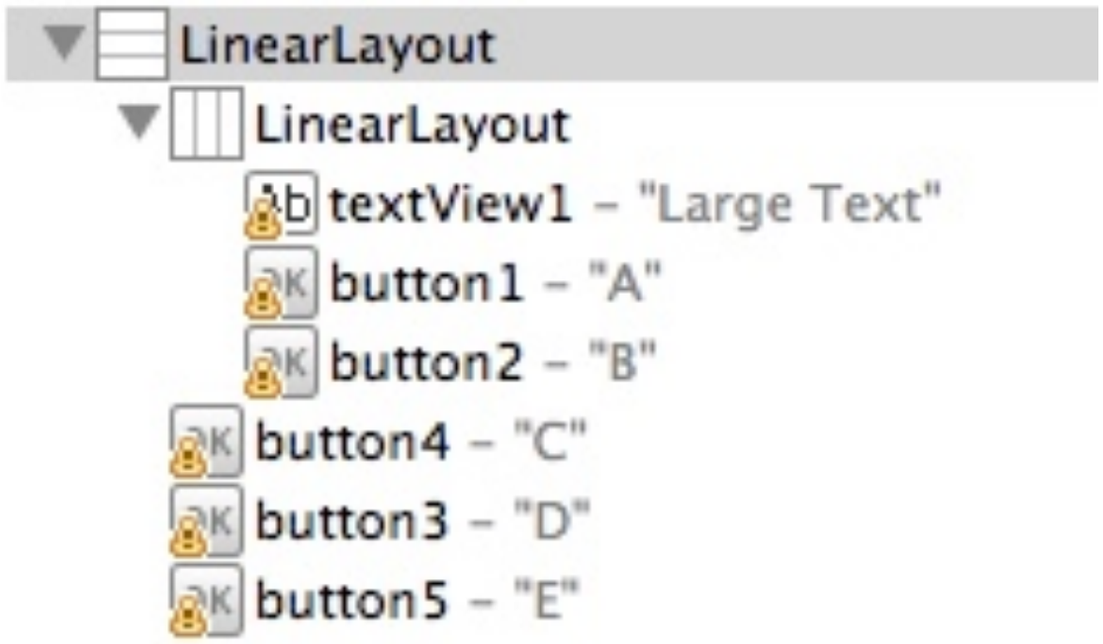
Just move widgets to where you want them to be

Their position is defined relative to other widgets

Eclipse Graphical editor sometimes gets confused if have lots of elements and move one

Nesting Layouts

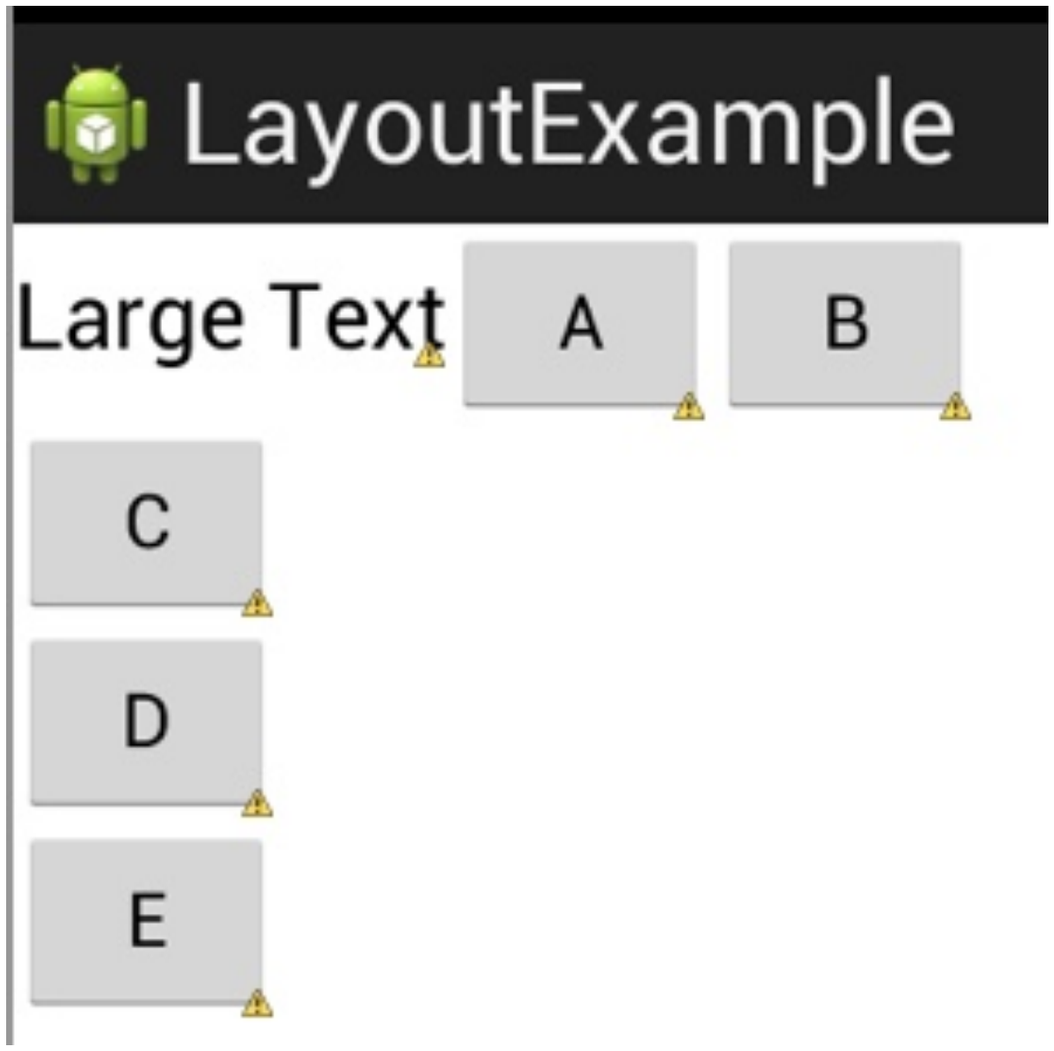




Don't nest too deep as this slows down rendering

Those Pesky Warnings

All text displayed on screen should be defined in strings



res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">LayoutExample</string>
    <string name="A">A</string>
    <string name="B">B</string>
    <string name="C">C</string>
    <string name="D">D</string>
    <string name="E">E</string>
    <string name="large">Large Text</string>
</resources>
```

res/layout/using_strings_linear.xml

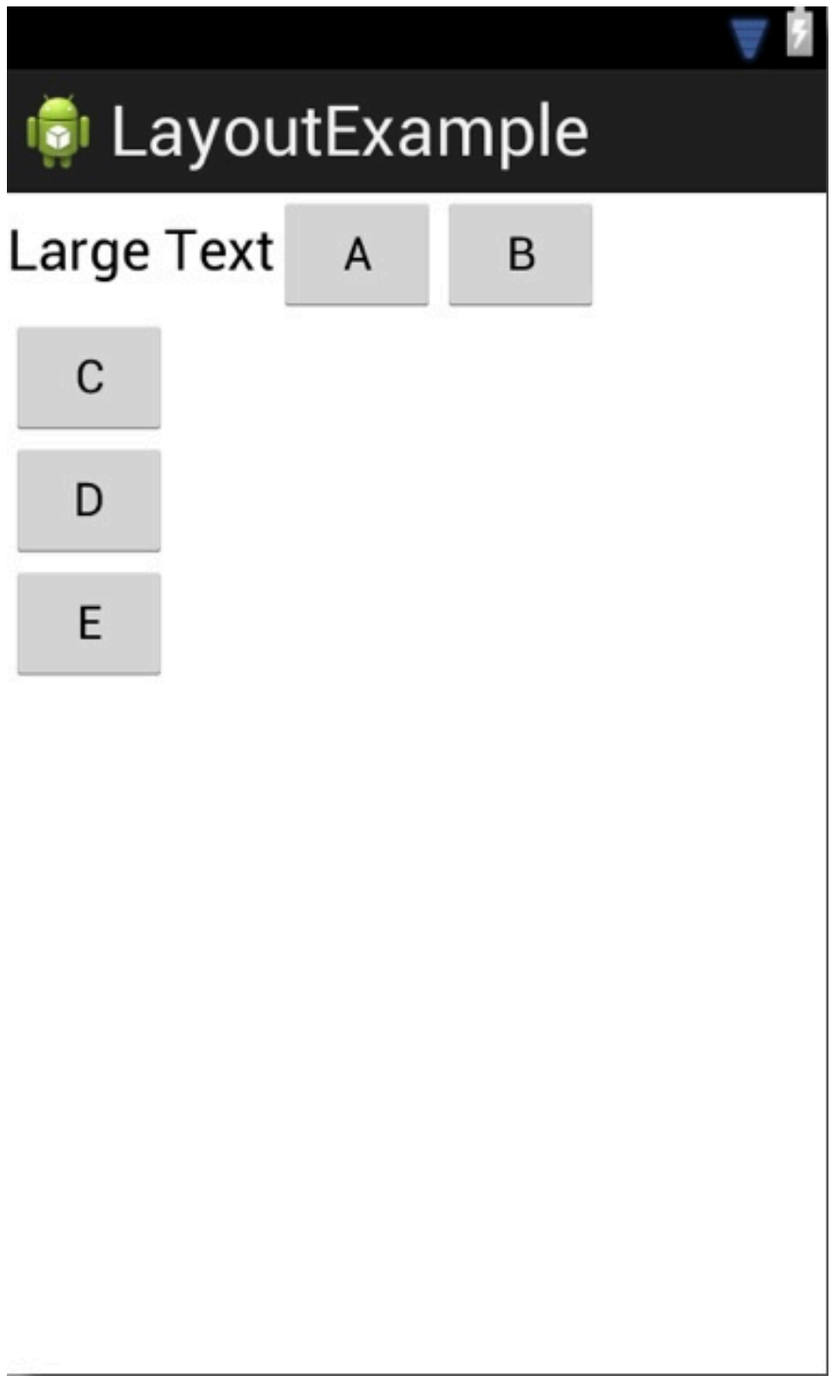
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/large"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/A" />

    <Button
        android:id="@+id/button2"
```

Layout Rendered



Why all the work?

strings.xml

<string name="large">Large Text</string>

layout.xml

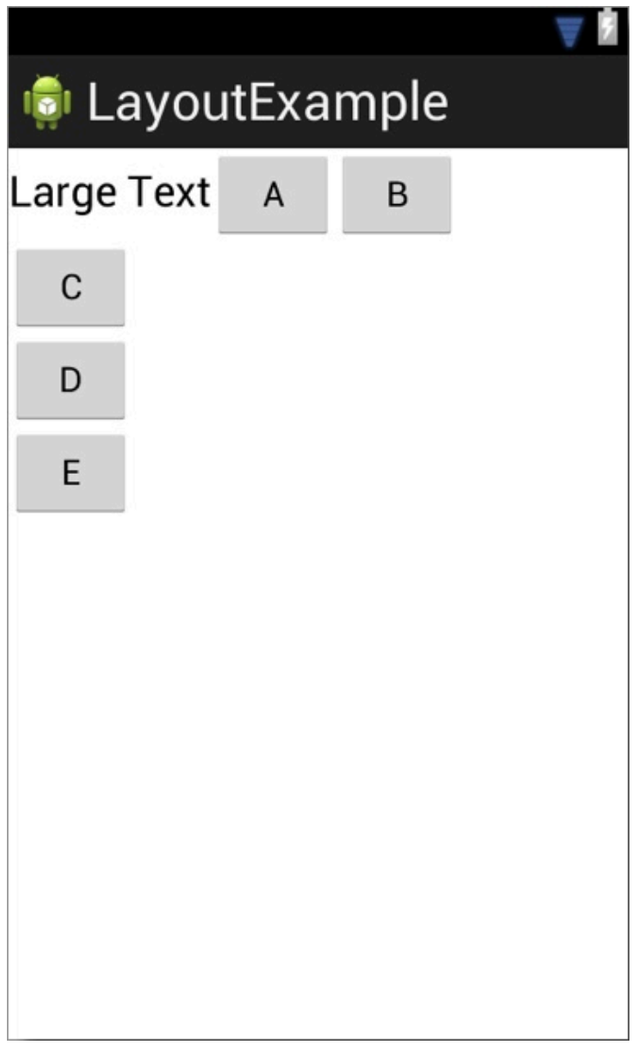
android:text="**@string/large**"

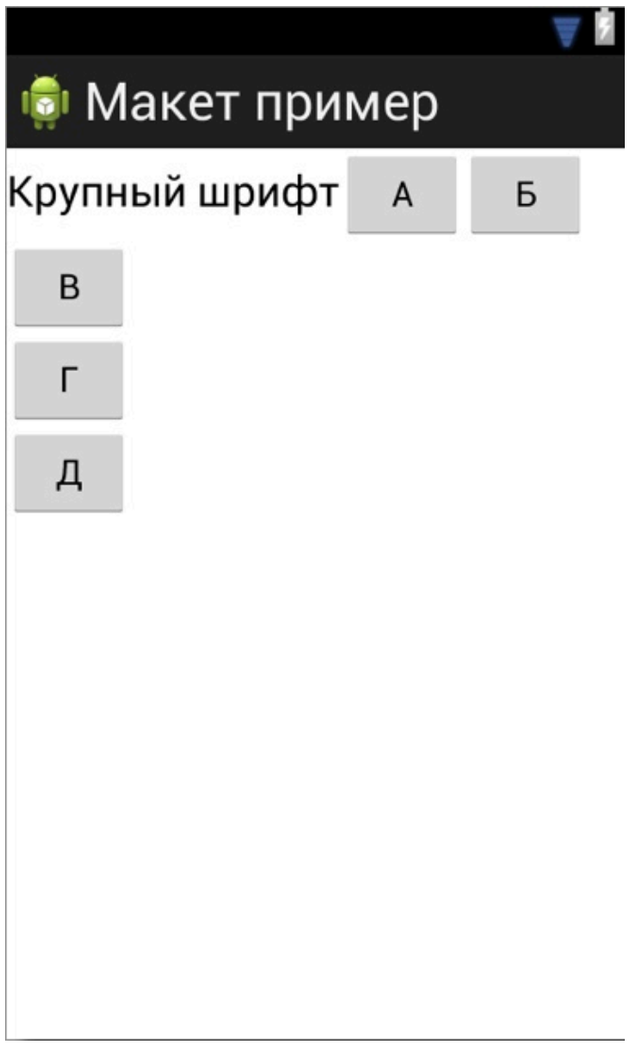
Verses

layout.xml

android:text="Large Text"

Localization





No Change in layout

Adding Langauges

For each language add file

res/values-xx/strings.xml xx - language code

with translation for each string

res/values-ru/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Макет пример</string>
    <string name="A">А</string>
    <string name="B">Б</string>
    <string name="C">В</string>
    <string name="D">Г</string>
    <string name="E">Д</string>
    <string name="large">Крупный шрифт</string>
</resources>
```

strings.xml - Name

You can name the file anything you like

strings.xml is default name generated by tools

You can have multiple files for strings

Common Pattern

Create same resource with same name in different folders

Each folder is marked for when you use

Example - layout

layout folder - contains default layouts to use
layout-land - contains layouts to use in landscape view

Layout Documentation with Examples

http://developer.android.com/guide/topics/ui/layout-objects.html

Saving Data

Permanent Data - Preference

Store Key value pairs

Key - string

Value
- boolean
- float
- int
- long
- string

Can have preferences for single activty

getPreferences(int mode)

Can share preference with other activities

getSharedPreferences(String name,int mode)

name allows you to group preferences

mode
- 0 = MODE_PRIVATE
- MODE_WORLD_READABLE
- MODE_WORLD_WRITEABLE
- MODE_MULTI_PROCESS (Android 3.0+)

But currently does not work across processes

Example

```
public class Calc extends Activity {
public static final String PREFS_NAME = "MyPrefsFile";
    protected void onCreate(Bundle state){
       super.onCreate(state);

       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
       boolean silent = settings.getBoolean("silentMode", false);
       setSilent(silent);
    }

    protected void onStop(){
       super.onStop();
       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
       SharedPreferences.Editor editor = settings.edit();
       editor.putBoolean("silentMode", mSilentMode);
       editor.commit();
    }
}
```


Example from http://code.google.com/android/devel/data/preferences.html. See http://developer.android.com/reference/android/app/Activity.html for an example of using getSharedPreferences()

Files - Internal Storage

Application can write/read files on phone

Cannot directly read files written by other application

Write a file

FileOutputStream openFileOutput(String name, int mode)

Creates file if it does not exist

mode

0 = MODE_PRIVATE

MODE_APPEND

MODE_WORLD_READABLE

MODE_WORLD_WRITEABLE

FileInputStream openFileInput(String name)

name can not contain path seperators

Static files

You can package static files with your application

Place file in res/raw/<mydatafile>

Generates resource id in R

Read file using

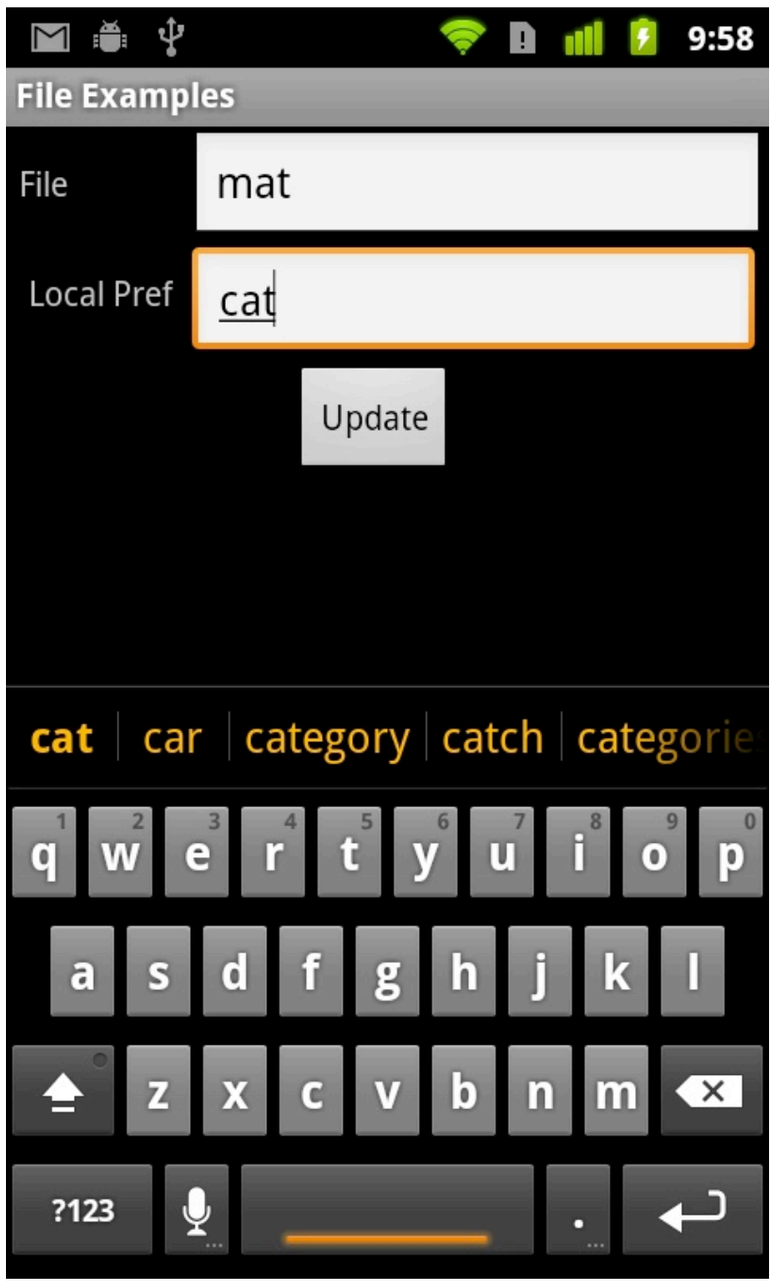
Resources.openRawResource (R.raw.mydatafile)

File Example

Saves data in local file

and

Uses local preference to store data



FileExample Structure

```
public class FileExamples extends Activity implements View.OnClickListener {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button update = (Button) findViewById(R.id.update);
        update.setOnClickListener(this);
        restoreData();
    }

    public void onClick(View v) {
        saveData();
    }
```

Getting Data from Files/Preference

```
private void restoreData() {
        String fileContents = readFile();
        EditText fileText = (EditText) this.findViewById(R.id.file);
        fileText.setText(fileContents);

        EditText preferenceText = (EditText) this
                        .findViewById(R.id.localPreference);
        SharedPreferences settings = getPreferences(MODE_PRIVATE);
        preferenceText.setText(settings.getString("setting", "No value"));
}
```

Read file

```
private String readFile() {
        String fileContents;
        try {
                InputStream file = new BufferedInputStream(
                                openFileInput("dataFile"));
                byte[] data = new byte[file.available()];
                file.read(data, 0, file.available());
                fileContents = new String(data);
                file.close();
        } catch (Exception noFile) {
                fileContents = "empty";
        }
        return fileContents;
}
```

Storing the Data

```
private void saveData() {
        EditText fileText = (EditText) this.findViewById(R.id.file);
        String fileContents = fileText.getText().toString();
        writeFile(fileContents);
        EditText preferenceText = (EditText) this
                        .findViewById(R.id.localPreference);
        String preferenceContents = preferenceText.getText().toString();
        SharedPreferences settings = getPreferences(MODE_PRIVATE);
        SharedPreferences.Editor editor = settings.edit();
        editor.putString("setting", preferenceContents);
        editor.commit();
}
```

Writing a File

```
private void writeFile(String fileContents) {
        try {
            OutputStream file = new BufferedOutputStream(openFileOutput(
                    "dataFile", MODE_PRIVATE));
            file.write(fileContents.getBytes());
            file.close();
        } catch (Exception noFile) {
        }
    }
}
```

File Modes

MODE_PRIVATE

MODE_APPEND

More Useful Methods

getFilesDir()

Gets the absolute path to directory where your internal files are saved.

getDir()

Creates (or opens an existing) directory within your internal storage space.

deleteFile()

Deletes a file saved on the internal storage.

fileList()

Returns an array of files currently saved by your application.

These are methods in parent class of Activty

What about files in Subdirectories

Use File object to create files & directories

```
File foo = getDir("foo", MODE_PRIVATE);
File bar = new File(foo, "bar");
OutputStream out = null;
   try {
     out = new BufferedOutputStream(new FileOutputStream(bar));
     out.write("dog".getBytes());
     out.close();
   }
   catch (Exception noFile) {
            Log.i("rew", "WRTIE error", noFile);
            noFile.printStackTrace();
     }
```

Java Streams & Readers, Writers

Streams

Deals with bytes

Readers, Writers

Deals in character - unicode

"dog".getBytes()

Converts Unicode to bytes

Uses Androids default