Lecture 05 (ch 6 of Pro Android 4)

Building UIs and Using Controls

CMSC 4303 Mobile Apps Programming

Hong K. Sung, Ph.D.
Department of Computer Science
University of Central Oklahoma

1

Content Providers (ch 4)

- A content provider is to abstract data into services.
- The content provider makes data sources look like data providers of web sites.
- Content provider abstraction is required only if you want to share data externally or between applications.
- For internal data access:
  - preferences: a set of key/value pair
  - files
  - SQLite: SQLite database
  - Network: through Internet via HTTP services

2

intents (ch 5)

- **purpose**: to invoke components
- components in Android
  - activities: UI components
  - services: background code
  - broadcast receivers: code that responds to broadcast messages
  - content provides: code that abstract data

3

UI Nomenclature

Table 6–1. *UI Nomenclature*

Term	Description
View, widget, control	Each of these represents a UI element. Examples include a button, a grid, a list, a window, a dialog box, and so on. The terms *view*, *widget*, and *control* are used interchangeably in this chapter.
Container	This is a view used to contain other views. For example, a grid can be considered a container because it contains cells, each of which is a view.
Layout	This is a visual arrangement of containers and views and can include other layouts.

4

an example

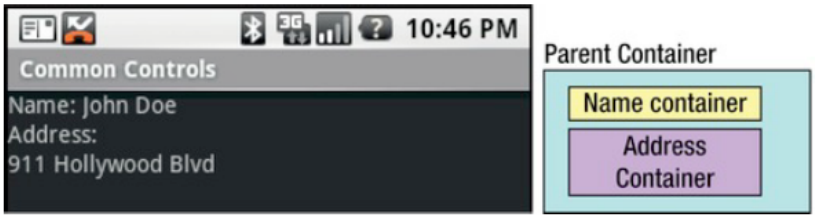


Figure 6–1. *The UI and layout of an activity*

The above example of UI can be done
1. completely in code (Listing 6-1)
2. completely in XML (Listing 6-2)
3. XML + code

5

building a UI in XML + code

Listing 6–3. *Creating a User Interface in XML with IDs*

```
<?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">
    <!-- NAME CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal" android:layout_width="fill_parent"
        android:layout_height="wrap_content">

            <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="@string/name_text" />

            <TextView android:id="@+id/nameValue"
        android:layout_width="wrap_content" android:layout_height="wrap_content" />

    </LinearLayout>
```

6

xml – cont'd

```
    <!-- ADDRESS CONTAINER -->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="wrap_content">

            <TextView android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/addr_text" />

            <TextView android:id="@+id/addrValue"
        android:layout_width="fill_parent" android:layout_height="wrap_content" />
    </LinearLayout>

</LinearLayout>
```

7

Listing 6–4. *strings.xml File for Listing 6–3*

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Common Controls</string>
    <string name="name_text">Name:</string>
    <string name="addr_text">Address:</string>
</resources>;
```

Listing 6–5. *Referring to Controls in Resources at Runtime*

```
setContentView(R.layout.main);

TextView nameValue = (TextView)findViewById(R.id.nameValue);
nameValue.setText("John Doe");
TextView addrValue = (TextView)findViewById(R.id.addrValue);
addrValue.setText("911 Hollywood Blvd.");
```

8

Text controls

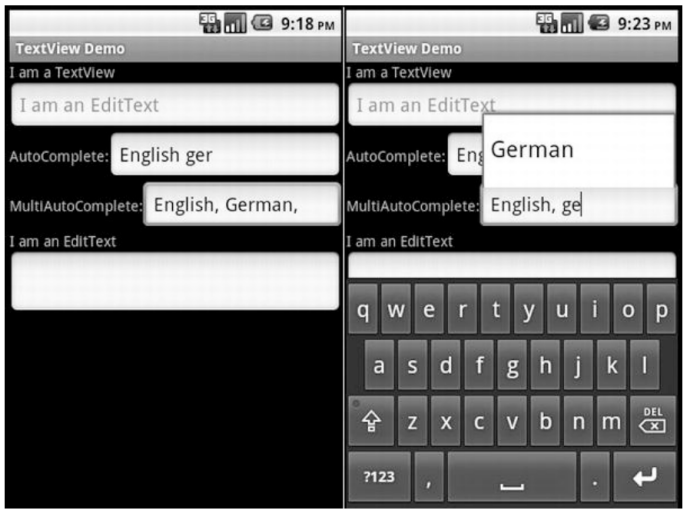


Figure 6–2. *Text controls in Android*

9

TextView

Listing 6–6. *Using Linkify on Text in a TextView*

```
TextView tv =(TextView)this.findViewById(R.id.tv);
tv.setAutoLinkMask(Linkify.ALL);
tv.setText("Please visit my website, http://www.androidbook.com
or email me at davemac327@gmail.com.");
```

```
<TextView android:id="@+id/tv" android:layout_width="wrap_content"
  android:layout_height="wrap_content"/>
```

10

EditText

- a subclass of TextView that allows for text editing.
- inputType property
  - singleLine (for a single line editing)
  - textMultiLine (for multiline typing)
- other properties
  - textAutoCorrect
  - textCapWords

11

AutoCompleteTextView

- a TextView with auto-complete functionality

Listing 6–7. *Using an AutoCompleteTextView Control*

```
<AutoCompleteTextView android:id="@+id/actv"
    android:layout_width="fill_parent"  android:layout_height="wrap_content" />


AutoCompleteTextView actv = (AutoCompleteTextView) this.findViewById(R.id.actv);

ArrayAdapter<String> aa = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line,
                new String[] {"English", "Hebrew", "Hindi", "Spanish",
                "German", "Greek" });

actv.setAdapter(aa);
```

12

MultiAutoCompleteTextView

- AutoCompleteTextView offers suggestions only for the entire text.
- MultiAutoCompleteTextView offers suggestions as the user types for each word.

Listing 6–8. *Using the MultiAutoCompleteTextView Control*

```
<MultiAutoCompleteTextView android:id="@+id/mactv"
    android:layout_width="fill_parent"  android:layout_height="wrap_content" />


MultiAutoCompleteTextView mactv = (MultiAutoCompleteTextView) this
                .findViewById(R.id.mactv);
ArrayAdapter<String> aa2 = new ArrayAdapter<String>(this,
                android.R.layout.simple_dropdown_item_1line,
new String[] {"English", "Hebrew", "Hindi", "Spanish", "German", "Greek" });

mactv.setAdapter(aa2);

mactv.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
```

13

Button controls

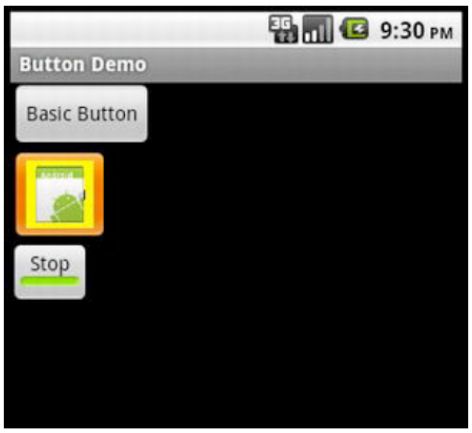


Figure 6–3. *Android button controls*

14

basic Button

Listing 6–9. *Handling Click Events on a Button*

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

Button button1 = (Button)this.findViewById(R.id.button1);
button1.setOnClickListener(new OnClickListener()
{
     public void onClick(View v)
     {
        Intent intent = new Intent(Intent.ACTION_VIEW,
                                   Uri.parse("http://www.androidbook.com"));
         startActivity(intent);
      }
 });
```

Listing 6–10. *Setting Up a Click Handler for a Button*

```
<Button   ...    android:onClick="myClickHandler"    ...  />

    public void myClickHandler(View target) {
        switch(target.getId()) {
        case R.id.button1:
        ...
```

15

ImageButton

Listing 6–11. *Using an* `ImageButton`

```
<ImageButton android:id="@+id/imageButton2"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:onClick="myClickHandler"
    android:src="@drawable/icon"  />

ImageButton imageButton2 = (ImageButton)this.findViewById(R.id.imageButton2);
imageButton2.setImageResource(R.drawable.icon); // dynamic change option
```

16

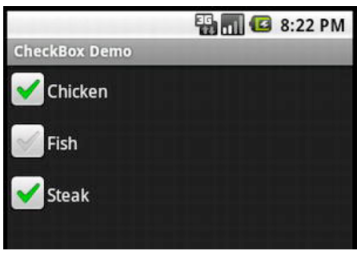
ToggleButton

Listing 6–13. *The Android* `ToggleButton`

```
<ToggleButton android:id="@+id/cctglBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Toggle Button"
        android:textOn="Stop"
        android:textOff="Run"/>
```

17

CheckBox



Listing 6–14. *Creating Check Boxes*

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical" android:layout_width="fill_parent"
        android:layout_height="fill_parent">

<CheckBox android:id="@+id/chickenCB"  android:text="Chicken" android:checked="true"
    android:layout_width="wrap_content" android:layout_height="wrap_content" />

<CheckBox android:id="@+id/fishCB"  android:text="Fish"
    android:layout_width="wrap_content" android:layout_height="wrap_content" />

<CheckBox android:id="@+id/steakCB"  android:text="Steak" android:checked="true"
    android:layout_width="wrap_content" android:layout_height="wrap_content" />

</LinearLayout>
```

18

Listing 6-15 Check Boxes in Code

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

            CheckBox fishCB = (CheckBox)findViewById(R.id.fishCB);

            if(fishCB.isChecked())
                fishCB.toggle();     // flips the checkbox to unchecked if it was
checked

            fishCB.setOnCheckedChangeListener(
                        new CompoundButton.OnCheckedChangeListener() {

                @Override
                public void onCheckedChanged(CompoundButton argo, boolean isChecked) {
                    Log.v("CheckBoxActivity", "The fish checkbox is now "
                            + (isChecked?"checked":"not checked"));
                }});
        }
}
```

19

RadioButton


```
<RadioGroup     android:id="@+id/rBtnGrp" android:layout_width="wrap_content"
            android:layout_height="wrap_content"  android:orientation="vertical" >

    <RadioButton     android:id="@+id/chRBtn" android:text="Chicken"
            android:layout_width="wrap_content"  android:layout_height="wrap_content"/>

    <RadioButton  android:id="@+id/fishRBtn" android:text="Fish" android:checked="true"
            android:layout_width="wrap_content"  android:layout_height="wrap_content"/>

    <RadioButton android:id="@+id/stkRBtn" android:text="Steak"
            android:layout_width="wrap_content"  android:layout_height="wrap_content"/>

</RadioGroup>
```

20

Listing 6-20 RadioButton Code

```
RadioGroup radGrp = (RadioGroup)findViewById(R.id.radGrp);

int checkedRadioButtonId = radGrp.getCheckedRadioButtonId();

radGrp.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(RadioGroup arg0, int id) {
        switch(id) {
        case -1:
            Log.v(TAG, "Choices cleared!");
            break;
        case R.id.chRBtn:
            Log.v(TAG, "Chose Chicken");
            break;
        case R.id.fishRBtn:
            Log.v(TAG, "Chose Fish");
            break;
        case R.id.stkRBtn:
            Log.v(TAG, "Chose Steak");
            break;
```

21

ImageView (to disaply an image)

```
<ImageView android:id="@+id/image1"
  android:layout_width="wrap_content"  android:layout_height="wrap_content"
  android:src="@drawable/icon" />

<ImageView android:id="@+id/image2"
  android:layout_width="125dip"  android:layout_height="25dip"
  android:src="#555555" />

<ImageView android:id="@+id/image3"
  android:layout_width="wrap_content"  android:layout_height="wrap_content" />

<ImageView android:id="@+id/image4"
  android:layout_width="wrap_content"  android:layout_height="wrap_content"
  android:src="@drawable/manatee02"
  android:scaleType="centerInside"
  android:maxWidth="35dip"  android:maxHeight="50dip"
  />
```

#rrggbb = color value

set image by program

22

Listing 6-21: ImageViews in code

```
ImageView imgView = (ImageView)findViewById(R.id.image3);

imgView.setImageResource( R.drawable.icon );

imgView.setImageBitmap(BitmapFactory.decodeResource(
            this.getResources(), R.drawable.manatee14) );

imgView.setImageDrawable(
            Drawable.createFromPath("/mnt/sdcard/dave2.jpg") );

imgView.setImageURI(Uri.parse("file://mnt/sdcard/dave2.jpg"));
```

23

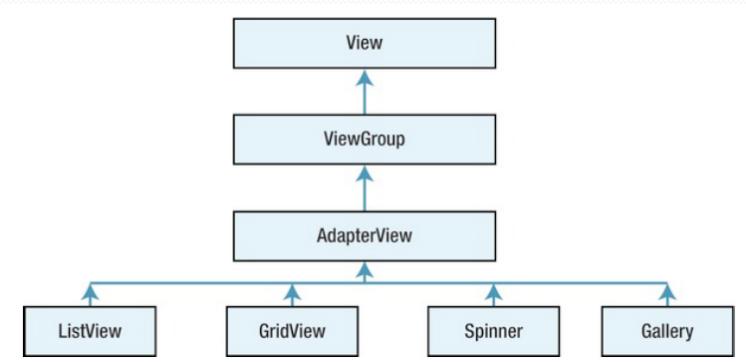
Date/Time Picker


```
<TextView android:id="@+id/dateDefault"
  android:layout_width="fill_parent" android:layout_height="wrap_content" />

<DatePicker android:id="@+id/datePicker"
  android:layout_width="wrap_content" android:layout_height="wrap_content" />

<TextView android:id="@+id/timeDefault"
  android:layout_width="fill_parent" android:layout_height="wrap_content" />

<TimePicker android:id="@+id/timePicker"
  android:layout_width="wrap_content" android:layout_height="wrap_content" />
```

24

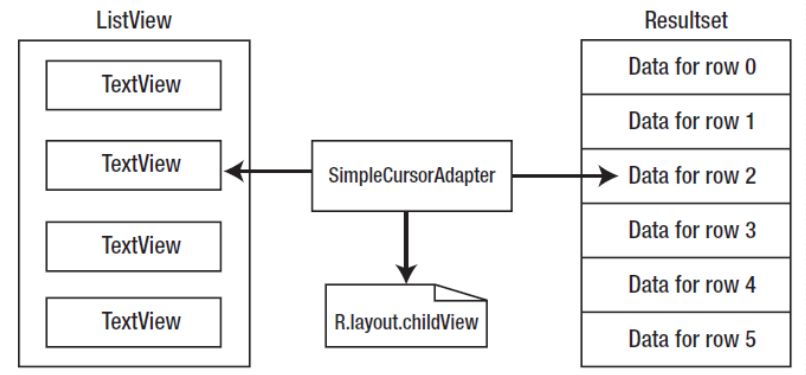
Adapters

- List controls are used to display collections of data.
- Instead of using a single type of control to manage both the display and the data, Android separates these two responsibilities into list controls and adapters.



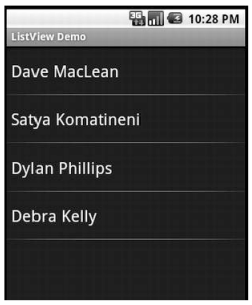
25

SimpleCursorAdapter



26

ListView




```
public class ListViewActivity extends ListActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        Cursor c = managedQuery(Contacts.CONTENT_URI,
                        null, null, null, Contacts.DISPLAY_NAME + " ASC");

        String[] cols = new String[] {Contacts.DISPLAY_NAME};
        int[]    views = new int[]   {android.R.id.text1};

        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
                        android.R.layout.simple_list_item_1,
                        c, cols, views);
        this.setListAdapter(adapter);
    }
}
```

27

deprecated

public final Cursor **managedQuery** (Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)

Parameters

uri	The URI of the content provider to query.
projection	List of columns to return.
selection	SQL WHERE clause.
selectionArgs	The arguments to selection, if any ?s are pesent
sortOrder	SQL ORDER BY clause.

Returns

The Cursor that was returned by query().

public **SimpleCursorAdapter** (Context context, int layout, Cursor c, String[] from, int[] to)

Parameters

context	The context where the ListView associated with this SimpleListItemFactory is running
layout	resource identifier of a layout file that defines the views for this list item. The layout file should include at least those named views defined in "to"
c	The database cursor. Can be null if the cursor is not available yet.
from	A list of column names representing the data to bind to the UI. Can be null if the cursor is not available yet.
to	The views that should display column in the "from" parameter. These should all be TextViews. The first N views in this list are given the values of the first N columns in the from parameter. Can be null if the cursor is not available yet.

28

ListView (up-to-date)

```
public class ListViewActivity extends ListActivity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        CursorLoader cloader = new CursorLoader(this, Contacts.CONTENT_URI,
                null, null, null, Contacts.DISPLAY_NAME + " ASC");
        Cursor c = cloader.loadInBackground();

        String[] cols = new String[]{Contacts.DISPLAY_NAME};
        int[]   views = new int[]   {android.R.id.text1};

        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
                android.R.layout.simple_list_item_1,
                c, cols, views, SimpleCursorAdapter.NO_SELECTION);
        this.setListAdapter(adapter);
    }
```

29

Permission: AndroidManifest.xml

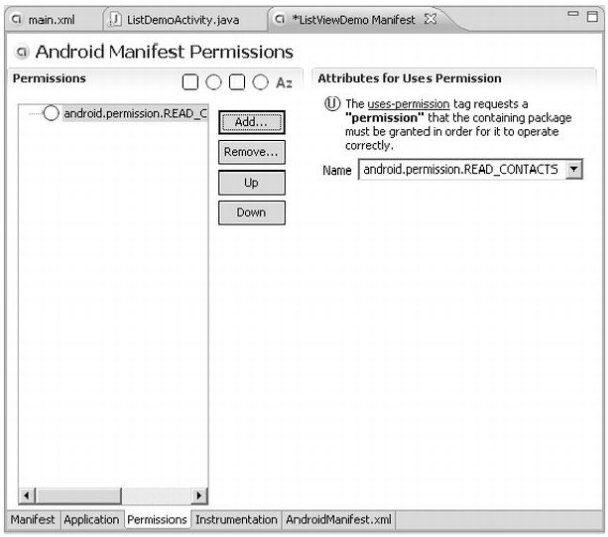


Figure 6–11. *Modifying `AndroidManifest.xml` so our application will run*

30

clickable items in ListView

```
public class ListViewActivity2 extends ListActivity implements OnItemClickListener
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        ListView lv = getListView();

        CursorLoader cloader = new CursorLoader(this, Contacts.CONTENT_URI,
                null, null, null, Contacts.DISPLAY_NAME + " ASC");
        Cursor c = cloader.loadInBackground();

        String[] cols = new String[]{Contacts.DISPLAY_NAME};
        int[]   views = new int[]   {android.R.id.text1};

        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
                android.R.layout.simple_list_item_1,
                c, cols, views, SimpleCursorAdapter.NO_SELECTION);
        this.setListAdapter(adapter);
        lv.setOnItemClickListener(this);
    }
```

31

click event handler

```
    public void onItemClick(AdapterView<?> adView, View target, int position, long id) {
        Log.v("ListViewActivity", "in onItemClick with " + ((TextView) target).getText() +
                ". Position = " + position + ". Id = " + id);
        Uri selectedPerson = ContentUris.withAppendedId(
                Contacts.CONTENT_URI, id);
        Intent intent = new Intent(Intent.ACTION_VIEW, selectedPerson);
        startActivity(intent);
    }
}
```

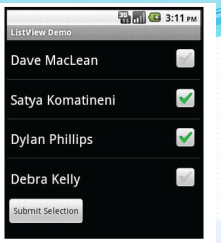
32

adding other control in ListView

```
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.list);

    lv = getListView();

    CursorLoader cloader = new CursorLoader(this, Contacts.CONTENT_URI,
            null, null, null, Contacts.DISPLAY_NAME + " ASC");
    cursor = cloader.loadInBackground();

    String[] cols = new String[]{Contacts.DISPLAY_NAME};
    idCol = cursor.getColumnIndex(Contacts._ID);
    nameCol = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
    timesContactedCol = cursor.getColumnIndex(Contacts.TIMES_CONTACTED);

    int[] views = new int[]{android.R.id.text1};

    SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
            android.R.layout.simple_list_item_multiple_choice,
            cursor, cols, views, SimpleCursorAdapter.NO_SELECTION);

    this.setListAdapter(adapter);

    lv.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
}
```

33

/res/layout/list.xml




```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is at /res/layout/list.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"  android:layout_height="fill_parent">

    <ListView android:id="@android:id/list"
        android:layout_width="fill_parent"  android:layout_height="0dip"
        android:layout_weight="1" />

    <Button android:id="@+id/button" android:onClick="doClick"
     android:layout_width="wrap_content"  android:layout_height="wrap_content"
     android:text="Submit Selection" />

</LinearLayout>
```

34

event handler

```
public void doClick(View view) {
    int count=lv.getCount();
    SparseBooleanArray viewItems = lv.getCheckedItemPositions();
    for(int i=0; i<count; i++) {
        if(viewItems.get(i)) {
            // CursorWrapper cw = (CursorWrapper) lv.getItemAtPosition(i);
            cursor.moveToPosition(i);
            long id = cursor.getLong(idCol);
            String name = cursor.getString(nameCol);
            int timesContacted = cursor.getInt(timesContactedCol);
            Log.v(TAG, name + " is checked. Times contacted = " + timesContacted +
                    ". Position = " + i + ". Id = " + id);
        }
    }
}
```

35

GridView

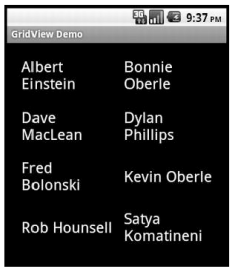
```
public class GridViewActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.gridview);

        GridView gv = (GridView)findViewById(R.id.gridview);

        Cursor c = managedQuery(Contacts.CONTENT_URI,
                        null, null, null, Contacts.DISPLAY_NAME);

        String[] cols = new String[] {Contacts.DISPLAY_NAME};
        int[]    views = new int[]      {android.R.id.text1};

        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
                android.R.layout.simple_list_item_1,
                c, cols, views);

        gv.setAdapter(adapter);
    }
}
```

36

/res/layout/gridview.xml

Listing 6–32. *Definition of a `GridView` in an XML Layout and Associated Java Code*

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is at /res/layout/gridview.xml -->
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10px"
    android:verticalSpacing="10px"
    android:horizontalSpacing="10px"
    android:numColumns="auto_fit"
    android:columnWidth="100px"
    android:stretchMode="columnWidth"
    android:gravity="center"
    />
```



37

Spinner

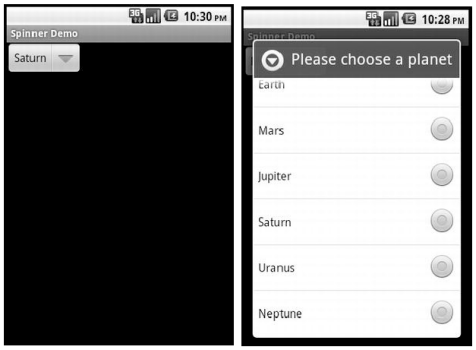
```
<Spinner
    android:id="@+id/spinner"  android:prompt="@string/spinnerprompt"
    android:layout_width="wrap_content"  android:layout_height="wrap_content" />
```

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

        Spinner spinner = (Spinner)findViewById(R.id.spinner);

        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
                R.array.planets, android.R.layout.simple_spinner_item);

        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

        spinner.setAdapter(adapter);
    }
}
```

38

/res/values/planets.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="planets">
    <item>Mercury</item>
    <item>Venus</item>
    <item>Earth</item>
    <item>Mars</item>
    <item>Jupiter</item>
    <item>Saturn</item>
    <item>Uranus</item>
    <item>Neptune</item>
  </string-array>
</resources>
```



39

Gallery

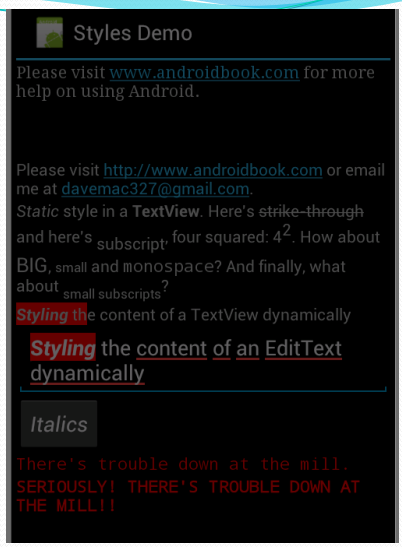
- The Gallery control is a horizontally scrollable list control that always focuses at the center of the list.
- The Gallery control is typically used to display images, so your adapter is likely going to be specialized for images.


```
<Gallery
    android:id="@+id/gallery"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
```

40

Style

- A style is a collection of View attributes that is given a name so you can refer to that collection by its name and assign that style by name to views.
- Spannable is a String that you can apply styles to.



41

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.styles);

    TextView tv =(TextView)this.findViewById(R.id.tv);
    tv.setAutoLinkMask(Linkify.ALL);
    tv.setText("Please visit http://www.androidbook.com or email me at davemac327@gmail.com.");

    TextView tv3 =(TextView)this.findViewById(R.id.tv3);
    tv3.setText("Styling the content of a TextView dynamically",
           TextView.BufferType.SPANNABLE);
    Spannable spn = (Spannable) tv3.getText();
    spn.setSpan(new BackgroundColorSpan(Color.RED), 0, 10,
                Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
    spn.setSpan(new StyleSpan(android.graphics.Typeface.BOLD_ITALIC),
                0, 7, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);

    EditText et =(EditText)this.findViewById(R.id.et);
    et.setText("Styling the content of an EditText dynamically");
    Spannable spn2 = (Spannable) et.getText();
    spn2.setSpan(new BackgroundColorSpan(Color.RED), 0, 7,
                Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
    spn2.setSpan(new StyleSpan(android.graphics.Typeface.BOLD_ITALIC),
                0, 7, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
}
```

42

Layout Managers

Layout Manager	Description
LinearLayout	Organizes its children either horizontally or vertically
TableLayout	Organizes its children in tabular form
RelativeLayout	Organizes its children relative to one another or to the parent
FrameLayout	Allows you to dynamically change the control(s) in the layout
GridLayout	Organizes its children in a grid arrangement

43

LinearLayout

- horizontally or vertically based on **orientation**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"  android:layout_height="wrap_content">

    <!-- add children here-->

</LinearLayout>
```

44

weight & gravity of Layout

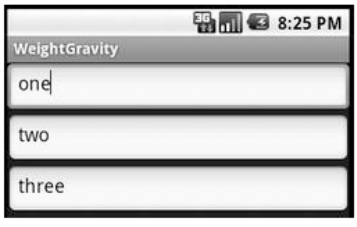
- weight: to assign size importance to a control relative to the other controls.
- gravity: for alignment (left, center, right, top, bottom, center_vertical, clip_horizontal)

45

weight & gravity: default

Listing 6–40. *Three Text Fields Arranged Vertically in a LinearLayout, Using Default Values for Weight and Gravity*

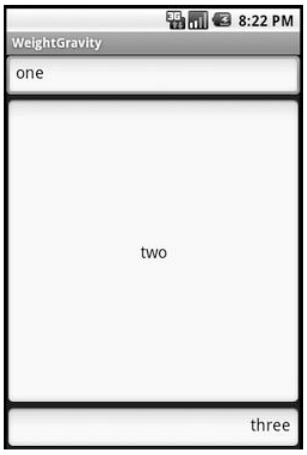
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="one"/>
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="two"/>
    <EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="three"/>
</LinearLayout>
```



46

weight & gravity

gravity
left
center
right



layout_weight
0
1
0

47

two 'gravity' controls

- android:gravity: used by the view
  - e.g., to center the text in the TextView
- android:layout_gravity: used by the container
  - e.g., to center the TextView in a LinearLayout


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <EditText android:layout_width="wrap_content" android:gravity="center"
    android:layout_height="wrap_content" android:text="one"
 android:layout_gravity="right"/>
</LinearLayout>
```

48

TableLayout

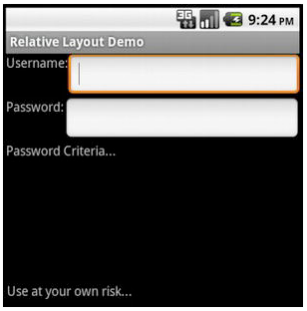
Android 4.0 adds GridLayout which is similar but is easier to use.


```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"  android:layout_height="fill_parent">

  <TableRow>
    <TextView android:text="First Name:"
        android:layout_width="wrap_content"  android:layout_height="wrap_content" />

    <EditText android:text="Edgar"
        android:layout_width="wrap_content"  android:layout_height="wrap_content" />
  </TableRow>

  <TableRow>
    <TextView android:text="Last Name:"
        android:layout_width="wrap_content"  android:layout_height="wrap_content" />

    <EditText android:text="Poe"
        android:layout_width="wrap_content"  android:layout_height="wrap_content" />
  </TableRow>

</TableLayout>
```

49

RelativeLayout

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

<TextView android:id="@+id/userNameLbl"
        android:layout_width="fill_parent"  android:layout_height="wrap_content"
        android:text="Username: "
        android:layout_alignParentTop="true" />

<EditText android:id="@+id/userNameText"
        android:layout_width="fill_parent"  android:layout_height="wrap_content"
        android:layout_toRightOf="@id/userNameLbl" />

 <TextView android:id="@+id/pwdLbl"
        android:layout_width="wrap_content"  android:layout_height="wrap_content"
        android:layout_below="@id/userNameText"
        android:text="Password: " />

 <EditText android:id="@+id/pwdText"
        android:layout_width="fill_parent"  android:layout_height="wrap_content"
        android:layout_toRightOf="@id/pwdLbl"
        android:layout_below="@id/userNameText" />

 <TextView android:id="@+id/pwdCriteria"
        android:layout_width="fill_parent"  android:layout_height="wrap_content"
        android:layout_below="@id/pwdText"
        android:text="Password Criteria... " />

 <TextView android:id="@+id/disclaimerLbl"
        android:layout_width="fill_parent"  android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Use at your own risk... " />

 </RelativeLayout>
```



50

FrameLayout

- mainly used to display a single item dynamically

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas
    android:id="@+id/frmLayout"
    android:layout_width="fill_parent"  an

     <ImageView
        android:id="@+id/oneImgView" andro
        android:scaleType="fitCenter"
        android:layout_width="fill_parent"
    <ImageView
        android:id="@+id/twoImgView" andro
        android:scaleType="fitCenter"
        android:layout_width="fill_parent"
        android:visibility="gone" />

</FrameLayout>
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.frame);

    one = (ImageView)findViewById(R.id.oneImgView);
    two = (ImageView)findViewById(R.id.twoImgView);

    one.setOnClickListener(new OnClickListener(){

        public void onClick(View view) {
            two.setVisibility(View.VISIBLE);

            view.setVisibility(View.GONE);
        }});

    two.setOnClickListener(new OnClickListener(){

        public void onClick(View view) {
            one.setVisibility(View.VISIBLE);

            view.setVisibility(View.GONE);
        }});
}
```

51