6

Android
Selection Widgets

Victor Matos
Cleveland State University

Notes are based on:

The Busy Coder's Guide to Android Development
by Mark L. Murphy

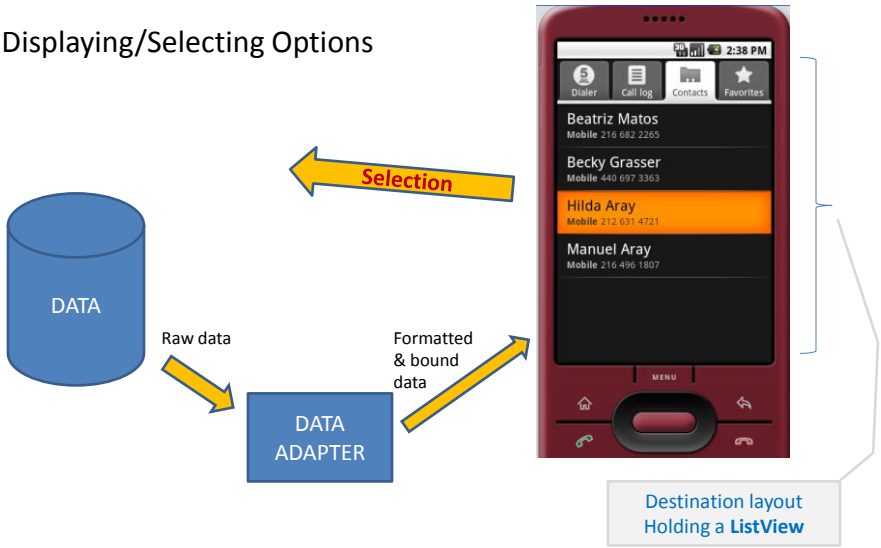
ISBN: 978-0-9816780-0-9
&
Android Developers
http://developer.android.com/index.html

Selection Widgets

- RadioButtons and CheckButtons are suitable for selecting from a *small* set of options.
- When the pool of choices is larger other widgets are more appropriate, those include classic UI controls such as: *listboxes, comboboxes, drop-down lists, picture galleries*, etc.
- Android offers a framework of *data adapters* that provide a common interface to selection lists ranging from static arrays to database contents.
- *Selection views* – widgets for presenting lists of choices – are handed an adapter to supply the actual choices.

Selection Widgets

Displaying/Selecting Options



Using ArrayAdapter

The easiest adapter to use is **ArrayAdapter** – all you need to do is wrap one of these around a Java array or java.util.List instance, and you have a fully functioning adapter:

```
String[] items={"this", "is", "a","really", "silly", "list"};

new ArrayAdapter<String>(this,
                         android.R.layout.simple_list_item_1,
                         items);
```

The *ArrayAdapter* constructor takes three parameters:

1. The *Context* to use (typically **this** will be your activity instance)
2. The resource ID of a *view* to use (such as the built-in system resource ***android.R.layout.simple_list_item_1*** as shown above)
3. The actual (source) array or list of ***items*** to show

Selection Widgets

Example 1: A simple list (1 of 4)

Instead of *Activity* we will use a *ListActivity* which is an Android class specializing in the use of ListViews.

```
<?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:id="@+id/selection"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff0000cc"
        android:textStyle="bold" />
    <!-- Here is the list. Since we are using a ListActivity, we  have to call it "@android:id/list" so  ListActivity will find it -->
    <ListView
        android:id="@android:id/list"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:drawSelectorOnTop="false" />

     <TextView android:id="@android:id/empty"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Empty set"  />
</LinearLayout>
```

Android's built-in list layout

Used on empty lists

Selection Widgets

Example 1 : A simple list (2 of 4)

```
package cis493.selectionwidgets;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

public class ArrayAdapterDemo extends ListActivity {

   TextView selection;
   String[] items = { "this", "is", "a", "really",
                      "really2", "really3","really4",
                      "really5", "silly", "list" };

   // next time try an empty list such as:
   // String[] items = {};
```

Data source

NOTE**: The **ListActivity** class is implicitly bound to an object identified by **@android:id/list

Selection Widgets

Example 1: A simple list (3 of 4)

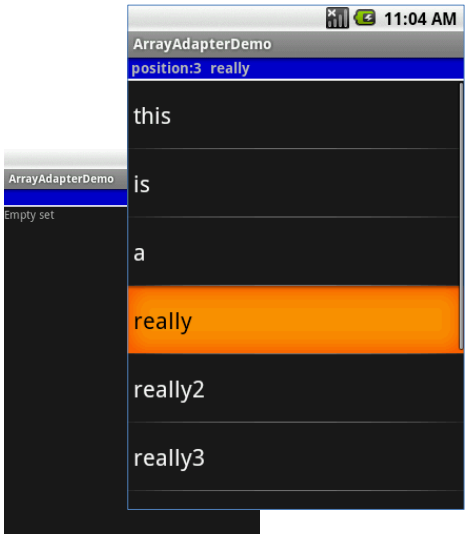
```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        setListAdapter(new ArrayAdapter<String>(
                       this,
                       android.R.layout.simple_list_item_1,
                       items));

        selection=(TextView)findViewById(R.id.selection);
    }

    @Override
    protected void onListItemClick(ListView l, View v,
                                   int position, long id) {
        super.onListItemClick(l, v, position, id);
        String text = " position:" + position + "  " + items[position];
        selection.setText(text);
    }

}
```

List adapter

Selection listener

Selection Widgets

Example 1: A simple list (4 of 4)



Selection seen by the listener

When you click here background flashes orange

Selection Widgets

Observations on Example1.

This example uses a number of predefined Android components.

1. In the XML layout we use a *ListView* widget called `android:id/list` (built-in definition using multiple lines, black background, light-gray separator line, horiz. scroll-bar)
2. Later in the setting of the ArrayAdapter we make a reference to `android.R.layout.simple_list_item_1` (details representation of a single entry in the list)

Android SDK includes a number of predefined layouts, they can be found in the folder: `C:\Android\platforms\android-1.6\data\res\layout`

*(See **Appendix A** for more on this issue)*

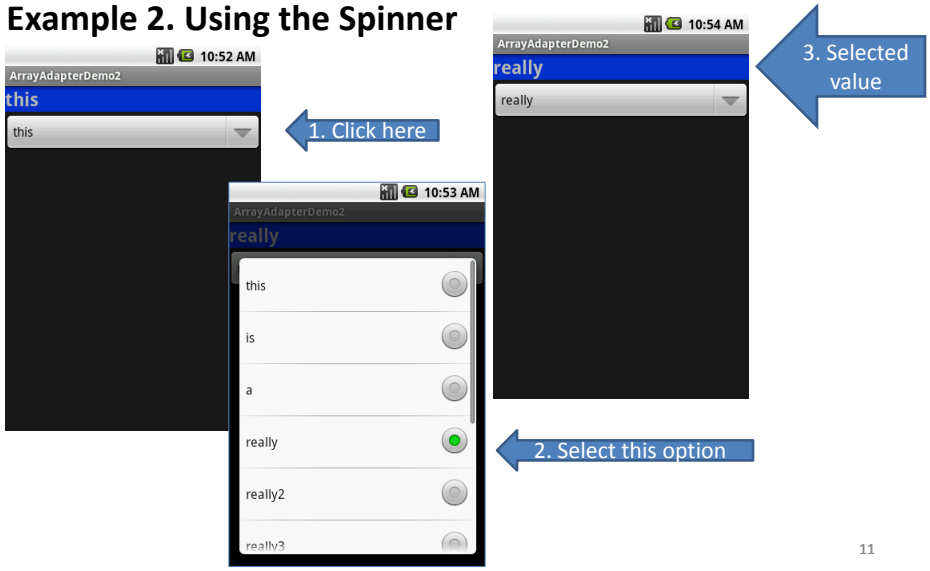
Selection Widgets

Spin Control

- In Android, the **Spinner** is the equivalent of the drop-down selector.
- Spinners have the same functionality of a ListView but take less space.
- As with ListView, you provide the adapter for linking data to child views using ***setAdapter()***
- Add a listener object to capture selections made from the list with ***setOnItemSelectedListener***().
- Use the ***setDropDownViewResource()*** method to supply the resource ID of the multi-line selection list view to use.

Selection Widgets

Example 2. Using the Spinner



Selection Widgets

Example 2. Using the Spinner

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/myLinearLayout"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/selection"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0033cc"
android:textSize="14pt"
android:textStyle="bold"
>
</TextView>
<Spinner
android:id="@+id/spinner"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
>
</Spinner>
</LinearLayout>
```

You choose the name

Selection Widgets

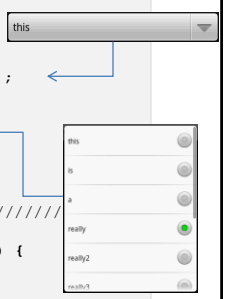
Example 2. Using the Spinner

```
package cis493.selectionwidgets;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;

public class ArrayAdapterDemo2 extends Activity
            implements AdapterView.OnItemSelectedListener {

   TextView selection;
   String[] items = { "this", "is", "a",
                      "really", "really2", "really3",
                      "really4", "really5", "silly", "list" };
```


Selection Widgets

Example 2. Using the Spinner

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

         selection = (TextView) findViewById(R.id.selection);

         Spinner spin = (Spinner) findViewById(R.id.spinner);
         spin.setOnItemSelectedListener(this);
         // set a clickable right push-button comboBox to show items
         ArrayAdapter<String> aa = new ArrayAdapter<String>(
                   this, android.R.layout.simple_spinner_item, items);
         // provide a particular design for the drop-down lines
         aa.setDropDownViewResource(
                   android.R.layout.simple_spinner_dropdown_item);
         // associate GUI spinner and adapter
         spin.setAdapter(aa);
    }
    // ////////////////////////////////////////////////////////////////////
    public void onItemSelected(
              AdapterView<?> parent, View v, int position, long id) {
         selection.setText(items[position]);
    }
    public void onNothingSelected(AdapterView<?> parent) {
         selection.setText("");
    }
}
```



Selection Widgets

GridView

GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

The grid items are automatically inserted to the layout using a ListAdapter.

Selection Widgets

GridView

Some properties used to determine the number of columns and their sizes:

- **android:numColumns** spells out how many columns there are, or, if you supply a value of auto_fit, Android will compute the number of columns based on available space and the properties listed below.
- **android:verticalSpacing** and its counterpart **android:horizontalSpacing** indicate how much whitespace there should be between items in the grid.
- **android:columnWidth** indicates how many pixels wide each column should be.
- **android:stretchMode** indicates, for grids with *auto_fit* for *android:numColumns*, what should happen for any available space not taken up by columns or spacing .

Selection Widgets

GridView

Example: Fitting the View
Suppose the screen is **320** pixels wide, and we have
android:columnWidth set to **100px** and
android:horizontalSpacing set to **5px**.

Three columns would use **310** pixels (three columns of 100 pixels and two whitespaces of 5 pixels).

With *android:stretchMode* set to *columnWidth*, the three columns will each expand by 3-4 pixels to use up the remaining 10 pixels.

With *android:stretchMode* set to *spacingWidth*, the two internal whitespaces will each grow by 5 pixels to consume the remaining 10 pixels.

Selection Widgets

Example 3. GridView

```
<?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:id="@+id/selection"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff0033cc"
             android:textSize="14pt"
             android:textStyle="bold"

        />
    <GridView
        android:id="@+id/grid"
        android:background="#ff0000ff"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:verticalSpacing="35px"
        android:horizontalSpacing="5px"
        android:numColumns="auto_fit"
        android:columnWidth="100px"
        android:stretchMode="columnWidth"
        android:gravity="center"
        />
</LinearLayout>
```

Selection Widgets

Example 3. GridView

```
package cis493.selectionwidgets;
// using a gridview
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.GridView;
import android.widget.TextView;

public class ArrayAdapterDemo3 extends Activity
                               implements AdapterView. OnItemClickListener {

    TextView selection;
    String[] items = { "this", "is", "a",
                       "really", "really2", "really3",
                       "really4", "really5", "silly", "list" };
```


Selection Widgets

Example 3. GridView

```
@Override
public void onCreate(Bundle icicle) {
    super.onCreate(icicle);
    setContentView(R.layout.main);
    selection = (TextView) findViewById(R.id.selection);

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

    ArrayAdapter<String> aa = new ArrayAdapter<String>(
                                  this,
                                  android.R.layout.simple_list_item_1,
                                  items );
    gv.setAdapter(aa);

    gv.setOnItemClickListener(this);
}

public void onItemClick(AdapterView<?> parent, View v,
                        int position, long id) {
    selection.setText(items[position]);
}

}// class
```

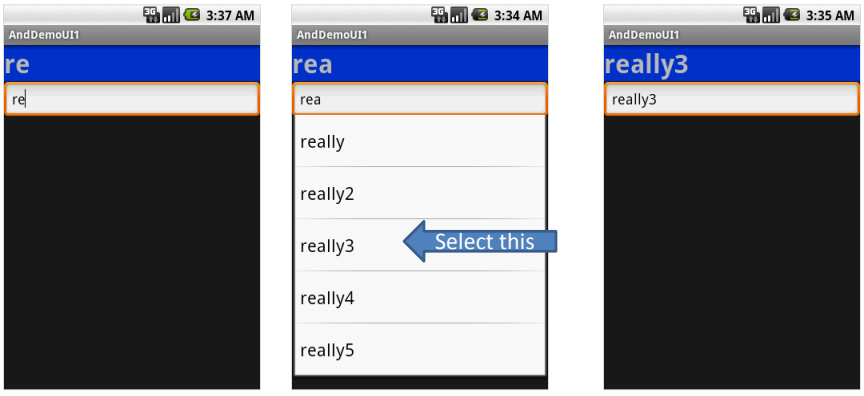
Selection Widgets

AutoCompleteTextView

- With **auto-completion**, as the user types, the text is treated as a prefix filter, comparing the entered text as a prefix against a list of candidates.
- Matches are shown in a *selection list* that, like with Spinner, folds down from the field.
- The user can either type out a *new entry* (e.g., something not in the list) or *choose an entry from the list* to be the value of the field.
- AutoCompleteTextView subclasses EditText, so you can configure all the standard look-and-feel aspects, such as font face and color.
- AutoCompleteTextView has a ***android:completionThreshold*** property, to indicate the minimum number of characters a user must enter before the list filtering begins.

Selection Widgets

AutoCompleteTextView



Selection Widgets

Example 4. AutoCompleteTextView

```
<?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:id="@+id/selection"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="#ff0033cc"
        android:textSize="14pt"
        android:textStyle="bold"
        />
    <AutoCompleteTextView
        android:id="@+id/edit"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:completionThreshold="3"/>
</LinearLayout>
```

Min. 3 chars to work

Selection Widgets

Example 4. AutoCompleteTextView

```
package cis493.selectionwidgets;

import android.app.Activity;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.TextView;

public class AndDemoUI1 extends Activity implements TextWatcher {

    TextView selection;

    AutoCompleteTextView edit;

    String[] items = { "this", "is", "a",
                       "really", "really2", "really3",
                       "really4", "really5", "silly", "list" };
```

Selection Widgets

Example 4. AutoCompleteTextView

```
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        selection = (TextView) findViewById(R.id.selection);

        edit = (AutoCompleteTextView) findViewById(R.id.edit);
        edit.addTextChangedListener(this);

        edit.setAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_dropdown_item_1line, items));
    }

    public void onTextChanged(CharSequence s, int start, int before, int count) {
        selection.setText(edit.getText());
    }

    public void beforeTextChanged(CharSequence s, int start,
                                  int count, int after) {
        // needed for interface, but not used
    }
    public void afterTextChanged(Editable s) {
        // needed for interface, but not used
    }
}
```


Selection Widgets

Gallery Widget

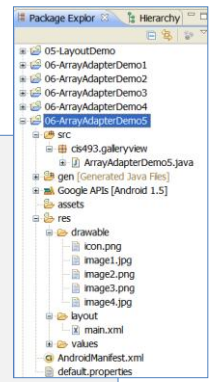
- The Gallery widget provides a set of options depicted as images.
- Image choices are offered on a contiguous horizontal mode, you may scroll across the image-set.

Selection Widgets

Gallery Widget - Example

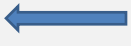
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="SDK1.5  /samples/.../view/Gallery1.java"
    />
    <TextView
    android:id="@+id/mySelection"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#ff0000ff"
    />
<Gallery
        android:id="@+id/myGallery"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="bottom"
/>
</LinearLayout>
```

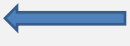
Selection Widgets



Gallery Widget - Example

```
package cis493.selectionwidgets;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemSelectedListener;


public class AndDemoUI1 extends Activity {
          // Using Gallery widget. G1 phone resolution: HVGA 320x480 px
          // code adapted from:
          // C:\Android\platforms\android-1.5\samples\ApiDemos\
          // src\com\example\android\apis\view\Galler1.java

          TextView mySelection;
          Gallery myGallery;
```

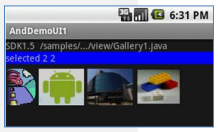




Selection Widgets

Gallery Widget - Example

```
@Override
public void onCreate(Bundle icicle) {
       super.onCreate(icicle);
       setContentView(R.layout.main);
       mySelection = (TextView) findViewById(R.id.mySelection);

       // Bind the gallery defined in the main.xml
       // Apply a new (customized) ImageAdapter to it.

       myGallery = (Gallery) findViewById(R.id.myGallery);

       myGallery.setAdapter(new ImageAdapter(this));
       myGallery.setOnItemSelectedListener(new OnItemSelectedListener() {

          @Override
          public void onItemSelected(AdapterView<?> arg0, View arg1,
                      int arg2, long arg3) {
                mySelection.setText(" selected option: " + arg2 );
          }

          @Override
          public void onNothingSelected(AdapterView<?> arg0) {
                mySelection.setText("Nothing selected");
          }

       });
}// onCreate
```



29

Selection Widgets

Gallery Widget - Example

```
public class ImageAdapter extends BaseAdapter {
        /** The parent context */
        private Context myContext;
        // Put some images to project-folder: /res/drawable/
        // format: jpg, gif, png, bmp, ...

        private int[] myImageIds = { R.drawable.image1, R.drawable.image2,
                                     R.drawable.image3, R.drawable.image4 };

        /** Simple Constructor saving the 'parent' context. */
        public ImageAdapter(Context c) {
                this.myContext = c;
        }

        // inherited abstract methods - must be implemented
        // Returns count of images, and individual IDs
        public int getCount() {
                return this.myImageIds.length;
        }

        public Object getItem(int position) {
                return position;
        }

        public long getItemId(int position) {
                return position;
        }
```

Selection Widgets

Gallery Widget - Example

```
    // Returns a new ImageView to be displayed,
    public View getView(int position, View convertView,
                        ViewGroup parent) {

        // Get a View to display image data
        ImageView iv = new ImageView(this.myContext);
        iv.setImageResource(this.myImageIds[position]);

        // Image should be scaled somehow
        //iv.setScaleType(ImageView.ScaleType.CENTER);
        //iv.setScaleType(ImageView.ScaleType.CENTER_CROP);
        //iv.setScaleType(ImageView.ScaleType.CENTER_INSIDE);
        //iv.setScaleType(ImageView.ScaleType.FIT_CENTER);
        //iv.setScaleType(ImageView.ScaleType.FIT_XY);
        iv.setScaleType(ImageView.ScaleType.FIT_END);

        // Set the Width & Height of the individual images
        iv.setLayoutParams(new Gallery.LayoutParams(95, 70));

        return iv;
        }
    }// ImageAdapter
}// class
```


Selection Widgets

GridView (again…)

A –perhaps-- more interesting version of the **GridView** control uses images instead of text.

The programmer must supply an **ImageAdapter** to indicate what to do when an individual image is selected/clicked.

The following example illustrates how to use this control.