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
Copyright © 2008-2009 CommonsWare, LLC.
ISBN: 978-0-9816780-0-9
&
Android Developers
http://developer.android.com/index.html

Selection Widgets

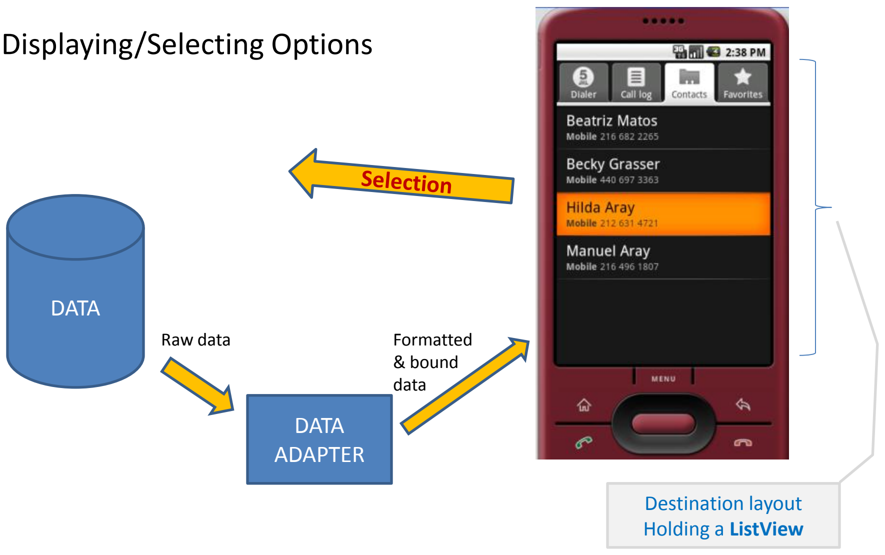
- RadioButtons and CheckButtons are suitable for selecting from a *small* set of options.
- When the pool of choices is larger other widgets are more appropriate, those include classic UI controls such as: *listboxes, comboboxes, drop-down lists, picture galleries*, etc.

- 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)

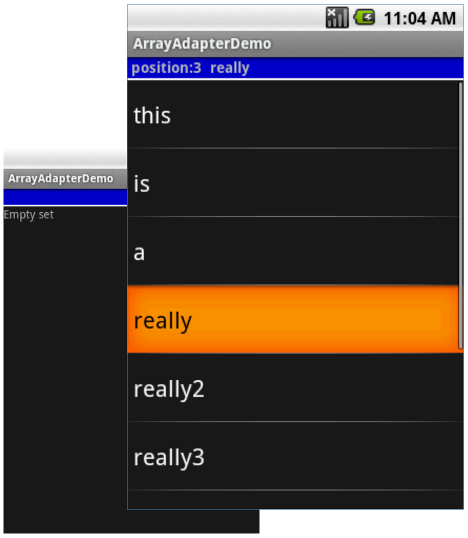
```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        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)



```
11:04 AM
ArrayAdapterDemo
position:3  really
this
is
a
really
really2
really3
```

```
ArrayAdapterDemo
Empty set
```

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)*

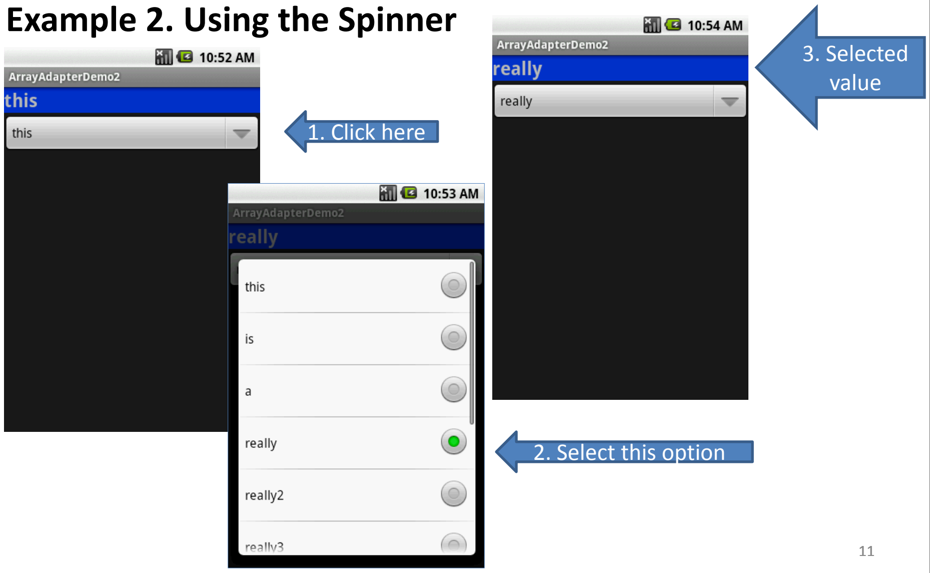
Selection Widgets

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



11

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

12

Selection Widgets

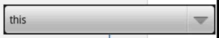
Example 2. Using the Spinner

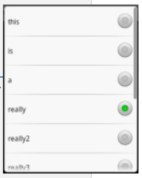
```
package cis493.selectionwidgets;

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("");
     }
}
```



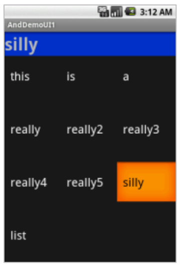


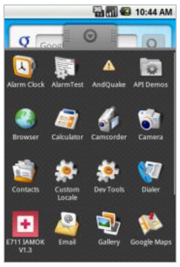
Selection Widgets

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).

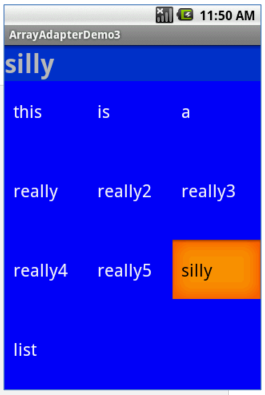
With *android:stretchMode* set to *columnWidth*, the three columns will each expand by 3-4 pixels to use up the remaining 10 pixels.

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>
```



silly

this is a
really really2 really3
really4 really5 silly
list

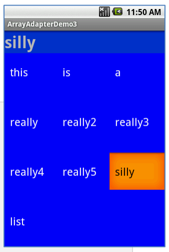
Selection Widgets

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
```

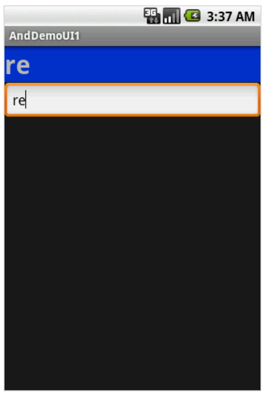
Selection Widgets

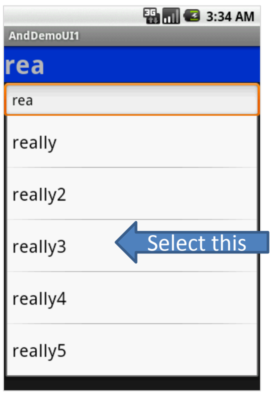
AutoCompleteTextView

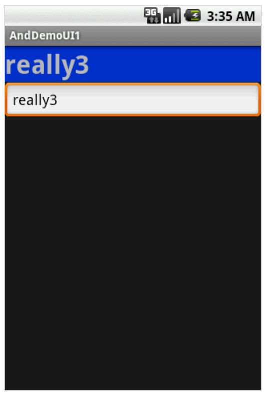
- With **auto-completion**, as the user types, the text is treated as a prefix filter, comparing the entered text as a prefix against a list of candidates.
- Matches are shown in a *selection list* that, like with Spinner, folds down from the field.
- The user can either type out a *new entry* (e.g., 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.

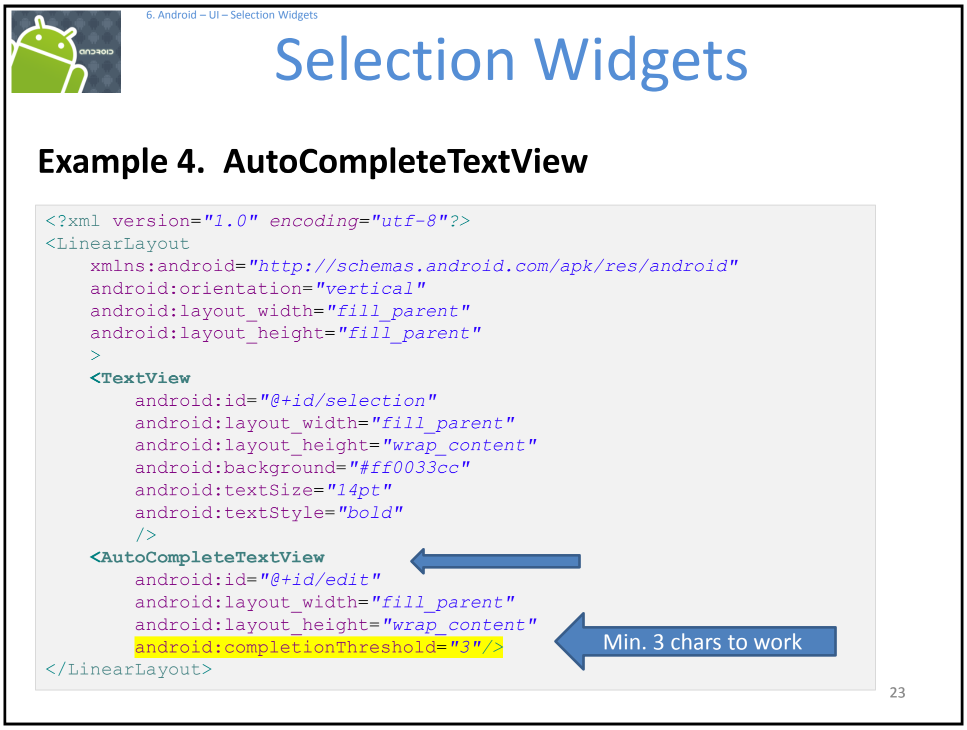
Selection Widgets

AutoCompleteTextView







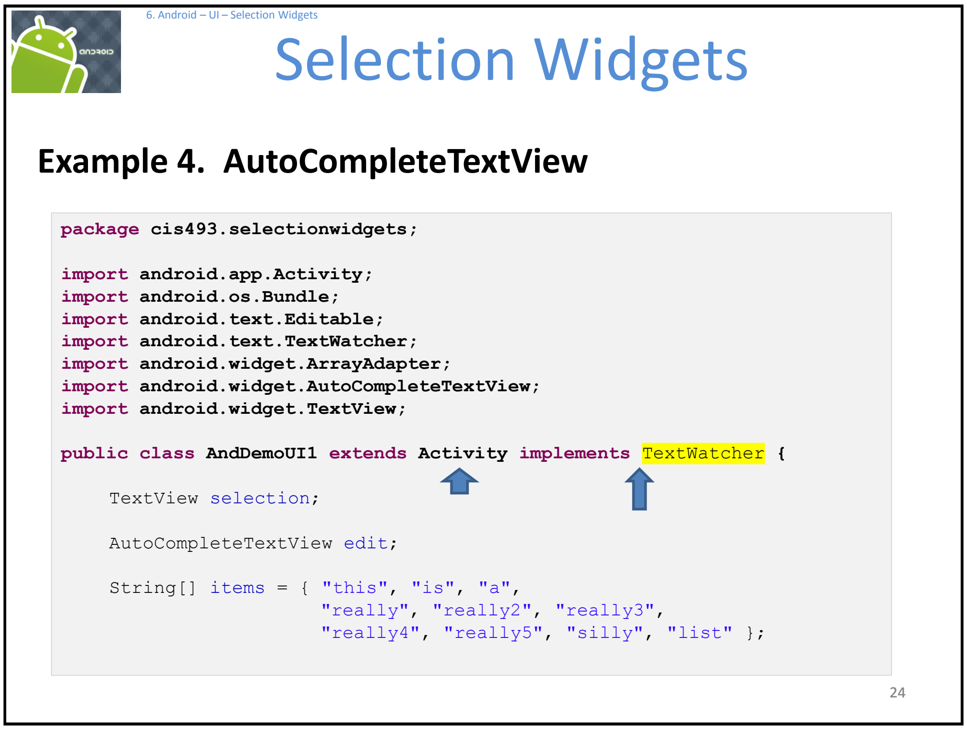


Selection Widgets

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

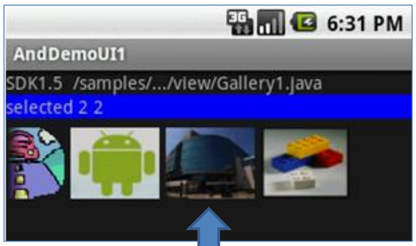
Example 4. AutoCompleteTextView

```
      @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.


6:31 PM
AndDemoUI1
SDK1.5 /samples/.../view/Gallery1.java
selected 2 2

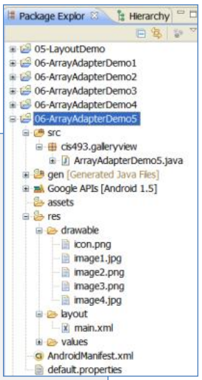
Selection Widgets

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



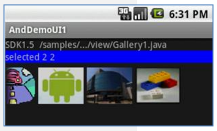
```
Package Explor   Hierarchy
05-LayoutDemo
06-ArrayAdapterDemo1
06-ArrayAdapterDemo2
06-ArrayAdapterDemo3
06-ArrayAdapterDemo4
06-ArrayAdapterDemo5
    src
        cis493.galleryview
            ArrayAdapterDemo5.java
    gen [Generated Java Files]
    Google APIs [Android 1.5]
    assets
    res
        drawable
            icon.png
            image1.jpg
            image2.png
            image3.png
            image4.jpg
        layout
            main.xml
        values
    AndroidManifest.xml
    default.properties
```

```
 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
```


6:31 PM
AndDemoUI1
SDK1.5 /samples/.../view/Gallery1.java
selected 2 2

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.

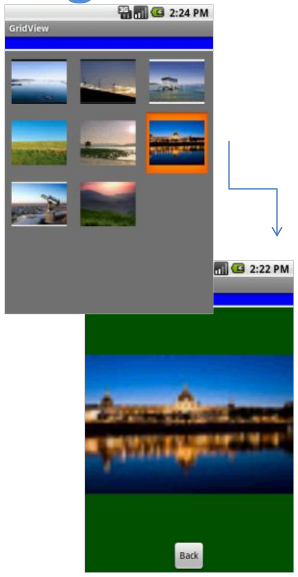
Selection Widgets

GridView (again…)

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

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

The following example illustrates how to use this control.

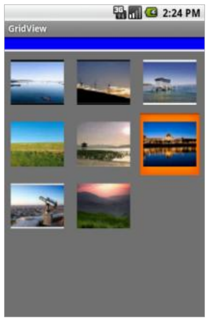


Selection Widgets

GridView (again…)

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:id="@+id/tvMsg"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#ff0000ff"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="3px"
    android:background="#ffffffff"
    />
<GridView
            android:id="@+id/gridview"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:numColumns="auto_fit"
            android:verticalSpacing="10dp"
            android:horizontalSpacing="10dp"
            android:columnWidth="90dp"
            android:stretchMode="columnWidth"
            android:gravity="center"
            android:background="#ff777777" />
</LinearLayout>
```

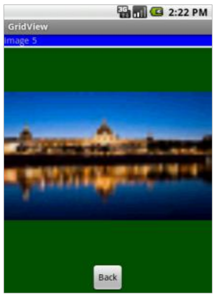


Selection Widgets

GridView (again…)

solo_picture.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"
    android:background="#ff005500"  >
<TextView
    android:id="@+id/tvSoloMsg"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#ff0000ff"  />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="3px"
    android:background="#ffffffff" />

<ImageView
           android:id="@+id/imgSoloPhoto"
           android:layout_width="fill_parent"
           android:layout_height="fill_parent"
           android:layout_gravity="center|fill"
           android:layout_weight="2"  />
<Button
           android:text="Back"
           android:id="@+id/btnBack"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:layout_gravity="center_horizontal" />
</LinearLayout>
```



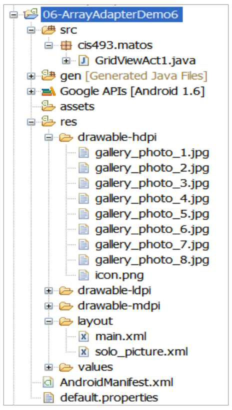
Selection Widgets

GridView (again…)

```
package cis493.matos;
/* ---------------------------------------------------------------------------
   References:
   Website on which you could make free thumbnails:
   http:  www.makeathumbnail.com/thumbnail.php
   ---------------------------------------------------------------------------
   GOAL: displaying a number of pictures in a GridView. Example taken from:
   http:  developer.android.com/guide/tutorials/views/hello-gridview.html
   ---------------------------------------------------------------------------
   Reference: http://developer.android.com/guide/practices/screens_support.html

   px       Pixels - corresponds to actual pixels on the screen.

   dp       Density-independent Pixels (dip) - an abstract unit that is based on the
            physical density of the screen. These units are relative to a
            160 dpi screen, so one dp is one pixel on a 160 dpi screen.
------------------------------------------------------------------------------
 */
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;
```

Selection Widgets

GridView (again…)



```
06-ArrayAdapterDemo6
  src
    cis493.matos
      GridViewAct1.java
  gen [Generated Java Files]
  Google APIs [Android 1.6]
  assets
  res
    drawable-hdpi
      gallery_photo_1.jpg
      gallery_photo_2.jpg
      gallery_photo_3.jpg
      gallery_photo_4.jpg
      gallery_photo_5.jpg
      gallery_photo_6.jpg
      gallery_photo_7.jpg
      gallery_photo_8.jpg
      icon.png
    drawable-ldpi
    drawable-mdpi
    layout
      main.xml
      solo_picture.xml
    values
  AndroidManifest.xml
  default.properties
```

```java
public class GridViewAct1 extends Activity implements OnItemClickListener {
    TextView tvMsg;
    GridView gridview;
    TextView tvSoloMsg;
    Integer[] mThumbIds;

    ImageView ivSoloPicture;
    Button btnBack;
    Bundle myMemoryBundle;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        myMemoryBundle = savedInstanceState;

        setContentView(R.layout.main);
        tvMsg = (TextView )findViewById(R.id.tvMsg);

        //initialize array of images with a few pictures
        mThumbIds = new Integer[]  {
                                    R.drawable.gallery_photo_1, R.drawable.gallery_photo_2,
                                    R.drawable.gallery_photo_3, R.drawable.gallery_photo_4,
                                    R.drawable.gallery_photo_5, R.drawable.gallery_photo_6,
                                    R.drawable.gallery_photo_7, R.drawable.gallery_photo_8
                                    };


        gridview = (GridView) findViewById(R.id.gridview);
        gridview.setAdapter(new ImageAdapter(this));
        gridview.setOnItemClickListener(this);

    }//onCreate
```

37

Selection Widgets

GridView (again…)

```java
    // this nested class could also be placed as a separated class
    public class ImageAdapter extends BaseAdapter {

            private Context mContext;
            public ImageAdapter(Context c) {
                        mContext = c;
                        }
            public int getCount() {
                        return mThumbIds.length;
                        }
            public Object getItem(int position) {
                        Toast.makeText(getApplicationContext(), "Pos: " + position, 1).show();
                        return null;
                        }
            public long getItemId(int position) {
                        return 0;
                        }
            // create a new ImageView for each item referenced by the Adapter
            public View getView(int position, View convertView, ViewGroup parent) {
                        ImageView imageView;
                        if (convertView == null) {
                                    // if it's not recycled, initialize some attributes
                                    imageView = new ImageView(mContext);
                                    imageView.setLayoutParams(new GridView.LayoutParams(85, 85));
                                    imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
                                    imageView.setPadding(8, 8, 8, 8);        }
                        else {
                                    imageView = (ImageView) convertView;
                                    }
                        imageView.setImageResource(mThumbIds[position]);
                        return imageView;
                        }
            }// ImageAdapter
```

38

Selection Widgets

GridView (again…)

```
        // a picture in the gallery view has been clicked
        @Override
        public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
                tvMsg.setText("Position: " + position +
                         "  R.drawable.gallery_photo_" + (position+1) );

                // show selected picture in an individual view
                showScreen2(position);
        }
        ////////////////////////////////////////////////////////////////////////////
        private void showScreen2(int position){
                // show the selected picture as a single frame
                setContentView(R.layout.solo_picture);
                tvSoloMsg = (TextView) findViewById(R.id.tvSoloMsg);
                ivSoloPicture = (ImageView) findViewById(R.id.imgSoloPhoto);
                tvSoloMsg.setText("image " + position);
                ivSoloPicture.setImageResource(mThumbIds[position]);

                btnBack = (Button) findViewById(R.id.btnBack);
                btnBack.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                // redraw the main screen beginning the whole app.
                                onCreate(myMemoryBundle);
                        }
                });

        }//showScreen2

        ////////////////////////////////////////////////////////////////////////////
}// GridViewAct1
```

Selection Widgets

Customized Lists

Android provides predefined row layouts for displaying simple lists. However, you may want more control in situations such as:

1. Not every row uses the same layout (e.g., some have one line of text, others have two)
2. You need to configure the widgets in the rows (e.g., different icons for different cases)

In those cases, the better option is to *create your own subclass of your desired Adapter.*

Selection Widgets

Customized Lists

In order to subclass your desired Adapter, you need to
1. override **getView(),** and
2. construct your rows yourself.

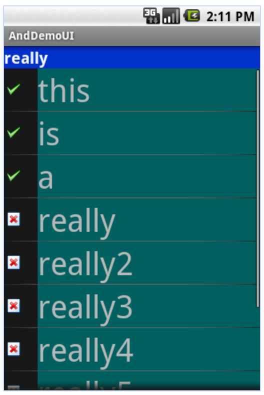
The *getView()* method is responsible for returning a View, representing the row for the supplied position of the adapter.

Example: Modify *getView()*, so we can have different icons for different rows in a list – in this case, one icon for short words and one for long words.

Selection Widgets

Customized Lists – Example: main.xml

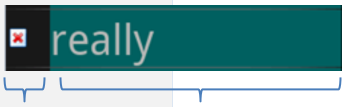
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/widget28"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/selection"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="#ff0033cc"
android:textSize="20px"
android:textStyle="bold"
android:textColor="#ffffffff"
>
</TextView>
<ListView
android:id="@android:id/list"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
</ListView>
</LinearLayout>
```


2:11 PM
AndDemoUI
really
✓ this
✓ is
✓ a
☒ really
☒ really2
☒ really3
☒ really4

Selection Widgets

Customized Lists – Example: myrow.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
>
    <ImageView
        android:id="@+id/icon"
        android:layout_width="22px"
        android:paddingLeft="2px"
        android:paddingRight="2px"
        android:paddingTop="2px"
        android:layout_height="wrap_content"
        android:src="@drawable/ok"
    />
    <TextView
        android:id="@+id/label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="40sp"
    />
</LinearLayout>
```



Selection Widgets

Customized Lists

```
import android.app.Activity;
import android.os.Bundle;
import android.app.ListActivity;
import android.view.View;
import android.view.ViewGroup;
import android.view.LayoutInflater;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
```

```
package cis493.demoui;

import ...

public class AndDemoUI extends ListActivity {
   TextView selection;
   String[] items = { "this", "is", "a", "really", "really2",
                      "really3", "really4", "really5", "silly", "list" };
   @Override
   public void onCreate(Bundle icicle) {
         super.onCreate(icicle);
         setContentView(R.layout.main);

         setListAdapter(new IconicAdapter(this));

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

   public void onListItemClick(ListView parent, View v, int position, long id) {
         selection.setText(items[position]);
   }
```

Selection Widgets

Customized Lists

```
  class IconicAdapter extends ArrayAdapter {
     Activity context;
     IconicAdapter(Activity context) {
          super(context, R.layout.myrow, items);
          this.context = context;
     }

     public View getView(int position, View convertView, ViewGroup parent) {
          LayoutInflater inflater = context.getLayoutInflater();
          View row = inflater.inflate(R.layout.myrow, null);
          TextView label = (TextView) row.findViewById(R.id.label);
           ImageView icon = (ImageView) row.findViewById(R.id.icon);
          label.setText(items[position]);

          if (items[position].length() > 4)
              icon.setImageResource(R.drawable.delete);
          else
              icon.setImageResource(R.drawable.ok);
          return (row);
     }//getView

  }//IconicAdapter

}//AndDemoUI
```

Selection Widgets

Customized Lists – LayoutInflater()

In this case, “inflation” means the act of converting an XML layout specification into the actual tree of View objects the XML represents.

An *ArrayAdapter* requires three arguments: current context, layout to show the output row, source data items (data to place in the rows).

The overridden **getView()** method inflates the layout by custom allocating *icons* and *text* taken from data source in the user designed row. Once assembled the View (row) is returned.

Selection Widgets

Questions ?

Selection Widgets

Appendix A. Android's Predefined Layouts

This is the definition of: ***simple_list_item_1 .*** It is just a TextView field named "**text1**" centered, of large font, and some padding.

```
<?xml version="1.0" encoding="utf-8" ?>

<!-- Copyright (C) 2006 The Android Open Source Project Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.   -->

<TextView xmlns:android="http://schemas.android.com/apk/res/android"
      android:id="@android:id/text1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:gravity="center_vertical"
      android:paddingLeft="6dip"
      android:minHeight="?android:attr/listPreferredItemHeight"
/>
```

Selection Widgets

Appendix A. Android's Predefined Layouts

This is the definition of: ***simple_spinner_dropdown_item.*** Other possible built-in spinner layout is: *simple_spinner_item*.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- legal comments removed !!! -->

<CheckedTextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    style="?android:attr/spinnerDropDownItemStyle"
    android:singleLine="true"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:ellipsize="marquee"
/>
```