Using Lists in Android

DT228/3

Dr. Susan McKeever

So far on the module:

XML Layouts/ Widgets

Event programming, Listeners..

Intents/ switching activities

Manifest file

Project structures

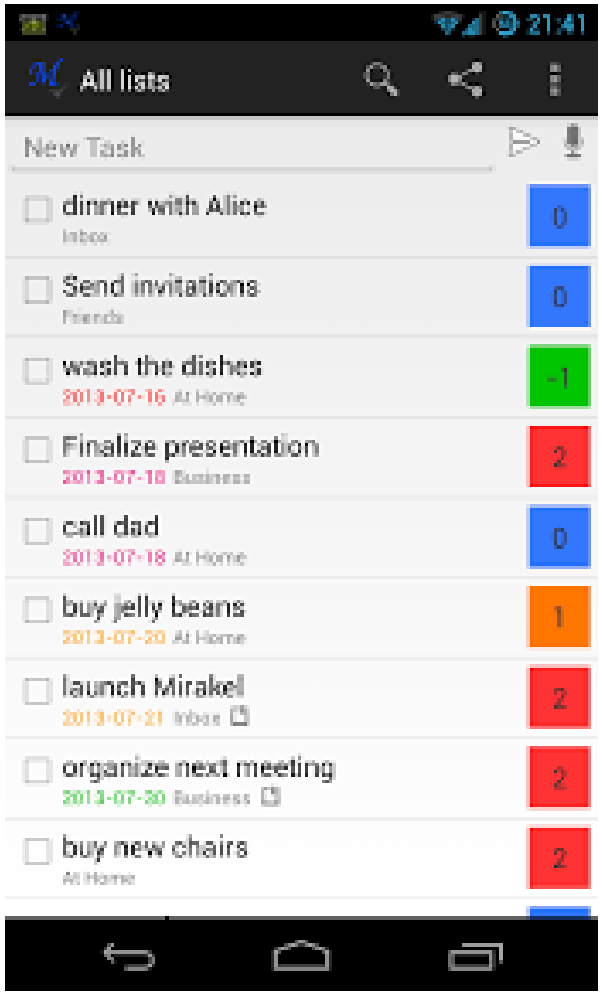
Containers

Resources..

Lists

- Very common in mobile apps
- Avoids data entry
- Interactions usually
  - via the toolbar
  - Direct item click for detail

•Check your own contacts list



Lists

What's needed to implement a list e.g. List of contacts on your phone

- **XML layout(s)**..
- **Data** for the list from somewhere.
- **current selection** if an item was clicked on
- **page scrolling** if a lot of data?
- Might want to **customise** your row. E.g. Images on each row? Checkboxes? Two rows per item? Etc

(Android helps with some of this…!)

Lists <ListView>

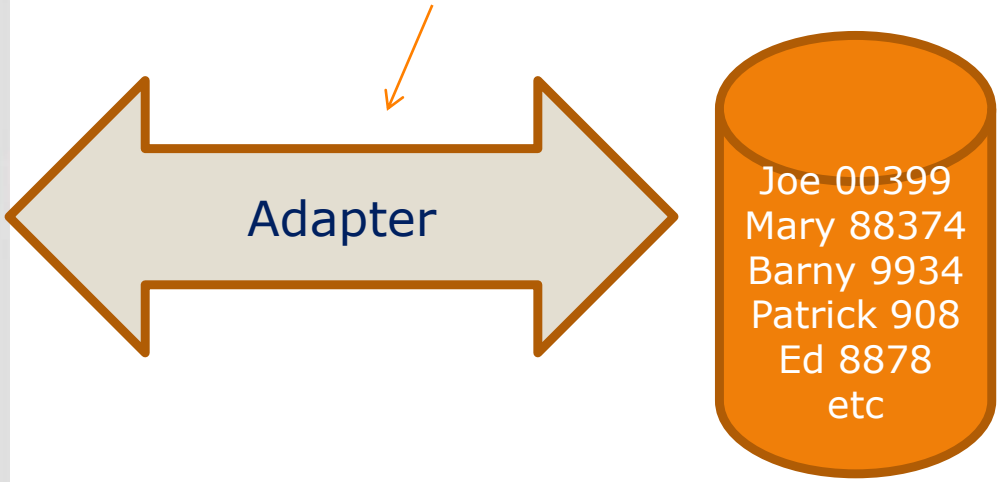
- **<ListView>** widget in your **XML layout** to create a list.

- Need data
  - Usually a query on a database..
  - Or.. if static...from an array[] {"joe", "mary"…}

- data adaptors handle the supply of data.

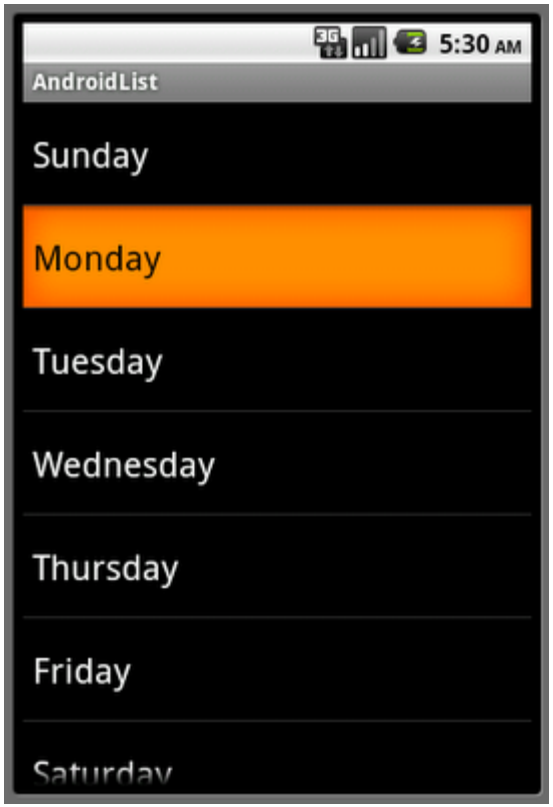
Adaptors to support lists

manage the supply of data and convert it into the display specified in the layout



- XMl layout has a <ListView> in it..
- Data is in a file Or array or a dB etc

Simplest Example of <ListView>



Displaying a list of Choices on a screen

Just line of plain text per list item

User can select one

Three types of list set ups

(1) Simple row layouts

To Display a simple one column list of text items

1 XML Screen layout (<Listview>)

Simple Example <ListView> In the XMl Layout

```
<ListView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:cacheColorHint="#00000000"
    android:id="@android:id/list">
</ListView>
```

- Just says that this will display a list of something..
- Only **one** <ListView> tag needed
- Note that the android:id is a special one: @android:id/list. Won't need to link from activity using findViewbyID

Simple Example <ListView>

- The activity (.java) needs to contain...

Use the list activity.. public class AndroidList extends ListActivity

Needs a source of data for the list.. e.g. String[] DayOfWeek =....

An adapter to bind the data to the list layout

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

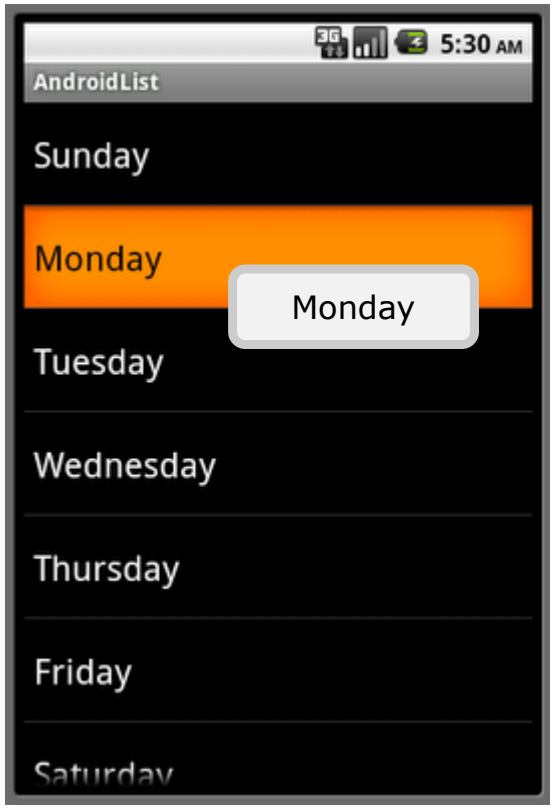
Various constructors for ArrayAdapter in the API..

This controls the appearance of the row - as one piece of text on it.
Android has loads of list row layouts e.g. simple_list_item_2 etc

Simple Example <ListView>

- Note: Ideally shouldn't hard code array data into your java code
  - String[] DayOfWeek = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};
- Can define arrays as external to your java (as a resource..
- Of if it's a list that's dynamic, will use a database.
- More on this later….

Clicking on a list item



Event programming
- usually need ?

No listener explicitly needs to be declared (if `ListActivity` used!)

Callback method:

```
onListItemClick(ListView l, View v, int position, long id)
```

Note these helpful parameters

Clicking on a list item

Event handler method
If using ListActivity
As subclass

```
public void onListItemClick(ListView l, View v, int position,
long id)
    {
      String selection = l.getItemAtPosition(position).toString();
      // whatever you want to do with the data returned.
      // getItemAtPosition() returns an object.. in this case, we know
      // it’s just one item, so we convert to a String
   }
}
```

So far on lists…

Now we know how to

- Display a list of simple text things
- Add in response to user click (OnListItemClick())
- So far, the rows are using a predefined row layout (using an Android row template)

Three types of list set ups

(1) Simple row layouts

To Display a simple one column list of text items

1 XML Screen layout *

Includes **<ListView>**

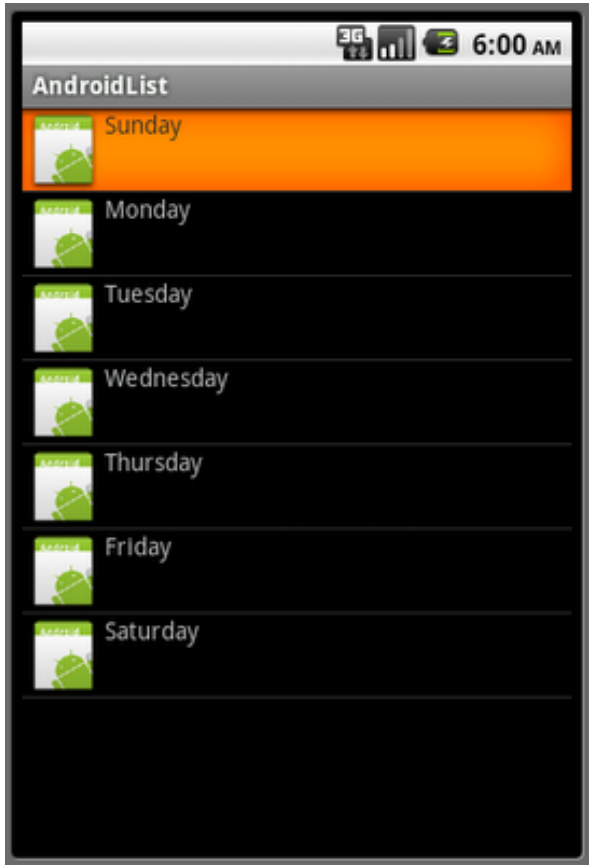
(2) Custom row layouts

To use your own row Layout

Also need a **Row.xml**

* Uses one of Android's predefined row layouts (eg. Android.r.layout. simple_list_item_1)

Custom row layout



If you want to display **more** than just one piece of text on each row ..

e.g.

An image on each row

Simple row layouts

To Display a simple one column list of text items

1 XML Screen layout Includes **<ListView>**

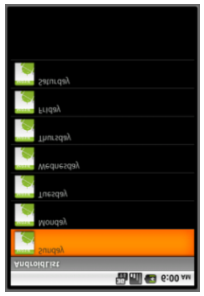
Uses one of Android's predefined row layouts (eg. Android.r.layout. simple_list_item_1)

Custom row layouts

To use your own row Layout

Need a second layout to define the row layout **Row.xml**

Example of a custom row layout

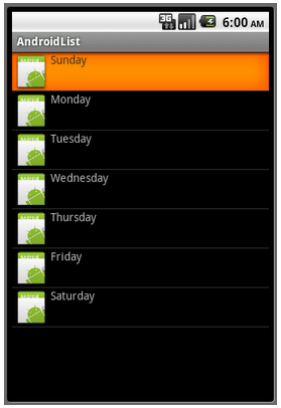


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns…blah ..>
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:orientation="horizontal">
  <ImageView
      android:id="@+id/icon"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/icon"/>
   <TextView
     android:id="@+id/weekofday"
     android:layout_width="wrap_content"
     droid:layout_height="wrap_content"/>

</LinearLayout>
```

Create a second XML layout for your horizontal row, e.g, Row.xml

Using your own row layout



To populate the a custom row layout, can still use an **existing off the shelf Adapter** for *simple* cases

```
setListAdapter(new ArrayAdapter<String>(this,
    R.layout.row, R.id.weekofday, DayOfWeek));
  }
}
```

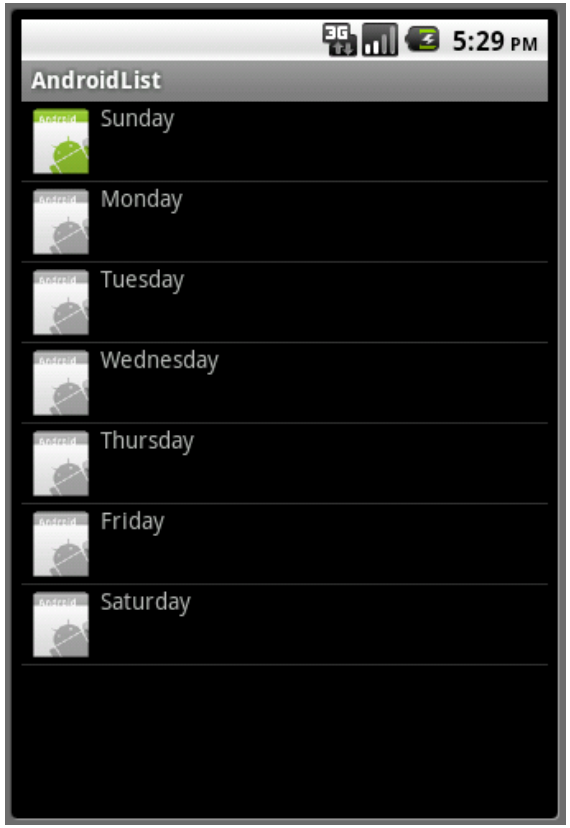
This is the XML file row.xml that holds your own layout

It replaces the predefined Simple_list_item_1

And you have to pick out the textivew "weekofday" where the list word will go…

Note: Different ArrayAdapter constructor to when we used Android predefined row layout

Third Scenario: different row layout depending on the data e.g. different icons/ check boxes, content etc



- You need to create your own adaptor
- We've just been using in-built ones for Android like arrayAdaptor
- E.g. Display green image if Sunday, else Grey image

Three types of list set ups

(1) Simple row layouts

To Display a simple one column list of text items

1 XML Screen layout

Includes **<ListView>**

(2) Custom row layouts

To use your own row Layout

Also need a **Row.xml**

(3) Custom adapters

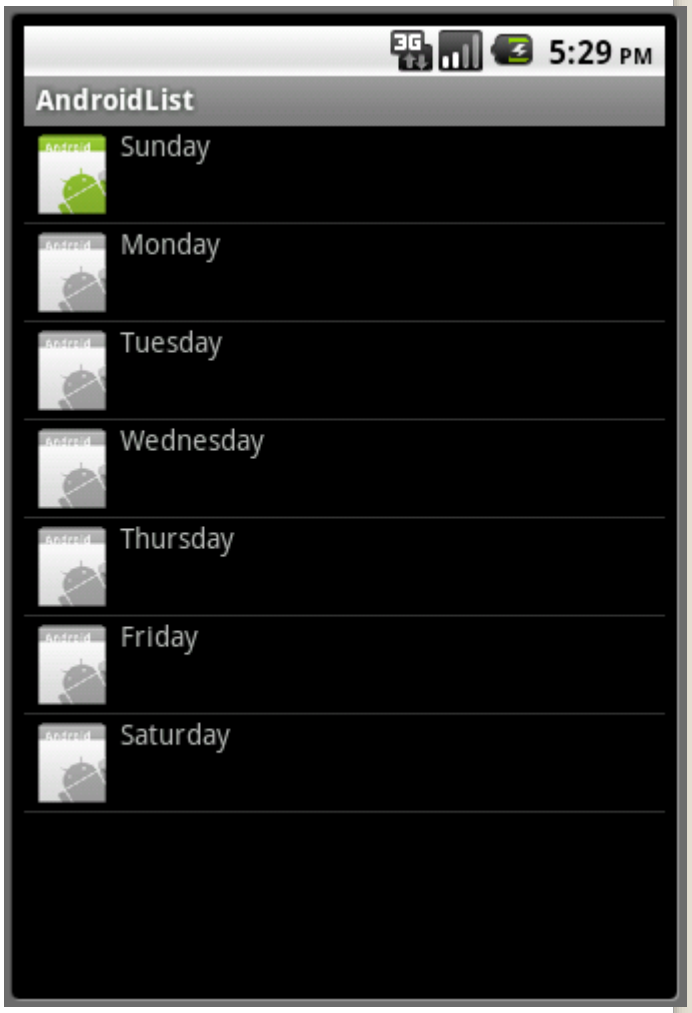
To make each row layout different (i.e. dynamic)

Also need to create your **own Adapter** (override getView and use convertView

Using custom adapters in a list

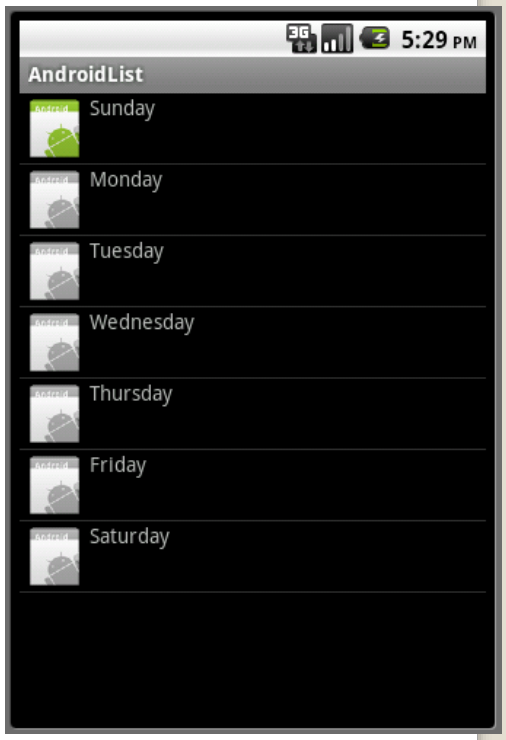
Making row layouts dymamic:

- E.g. a list of days. If it's "Sunday" different icon. Need to get inside the row inflation to control this
- To create a custom adaptor, extend an existing adapter
- E.g. ArrayAdapter/ CursorAdapter/ BaseAdapter



Using dynamic row layouts in a list

Inside your adapter..



- GetView() is called for *each* row in (most!) Android adapters
- **Override** the getView() method to format your own row in your way
- "Inflation" is getting at the XML for the row manually.. and making it usable
- Your custom adaptor typically an **inner class** inside the list activity

DynamicRows.java etc.

Custom adapters for custom lists

```
class MyList extends ListActivity
    {
        // Outer class: Data source declared, onCreate(),
        adapter set on the list (setListApapter) etc

        class MyCustomAdapter extends SomeExistingAdapter
        {
            //Inner Class: Gets the data source moved
                into rows.. done in here   -

            // getView method from parent adapter must be
                overridden
        }


    }
```

Custom adapter (using arrays) - outline

```
public class MyCustomAdapter extends ArrayAdapter<String>
{
// Constructor
  public MyCustomAdapter(Context context, int rowLayoutId,
           String[] myArrayData)
  {
    super(context, rowLayoutID, myArrayData);
   }

  @Override
  public View getView(int position, View convertView,
        ViewGroup parent)
  {
      View row;

      LayoutInflater inflater=getLayoutInflater();
      row=inflater.inflate(R.layout.row, parent, false);

         // Then, format the parts of your row layout that need
         formatting; Note - to connect to a widget ID in your row
         layout, use row.findViewByID

      return row;
 }
}
```

```
public View getView(int position, View convertView, ViewGroup parent)
{

 // TODO Auto-generated method stub
 //return super.getView(position, convertView, parent);

    LayoutInflater inflater=getLayoutInflater();
    View row=inflater.inflate(R.layout.row, parent, false);
    TextView label=(TextView)row.findViewById(R.id.weekofday);
    label.setText(DayOfWeek[position]);
    ImageView icon=(ImageView)row.findViewById(R.id.icon);

    if (DayOfWeek[position]=="Sunday")
    {
            icon.setImageResource(R.drawable.icon);
    }
    else
    {
        icon.setImageResource(R.drawable.icongray);
    }
   return row;
  }
```

To use the custom adapter in the activity:

```
setListAdapter(new MyCustomAdapter
                    (MyListClass.this,
                    rowlayoutID,
                    arrayObject));
```

Instead of the previous simple list one column example:

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

To increase efficiency for custom adaptors

If creating your own adaptor for managing your list:

- Very expensive to manually "inflate" each row as you go along.
- If have already scrolled through rows, can avoid having to "re-inflate" them all the time
- ConvertView is something Android lists do to avoid creating huge lists at a time
  - Recycle a row if it was created before
  - As you scroll, list items no longer visible are recycled to create a pool of list items
- In Getview() .. Check if any rows in convertview (i.e. recycled items to be displayed), otherwise, inflate the row if necessary.

Using ConvertView

Just Declare that you want to use convertView to manage your
Rows.. And then check if there are any available

If not.. Go ahead and inflate a new row

```
View row = convertView;

 if(row==null){
       LayoutInflater inflater=getLayoutInflater();
       row=inflater.inflate(R.layout.row, parent,
                                          false);
}
```

If list is empty? (no rows to display?)

In your main layout

```
<ListView..

Etc  />

<TextView android:id="@android:id/empty"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#FF0000"
            android:text="No data"/>
```

What we covered

Simple row layouts

To Display a simple one column list of text items

1 XML Screen layout

Includes **<ListView>**

Custom row layouts

To use your own row Layout

Also need a **Row.xml**

Custom adapters

To make each row layout different (i.e. dynamic)

Also need to create your **own Adapter** (override getView and use convertView

DDMS..Finding run time errors

- LogCAT
- Fatal errors etc