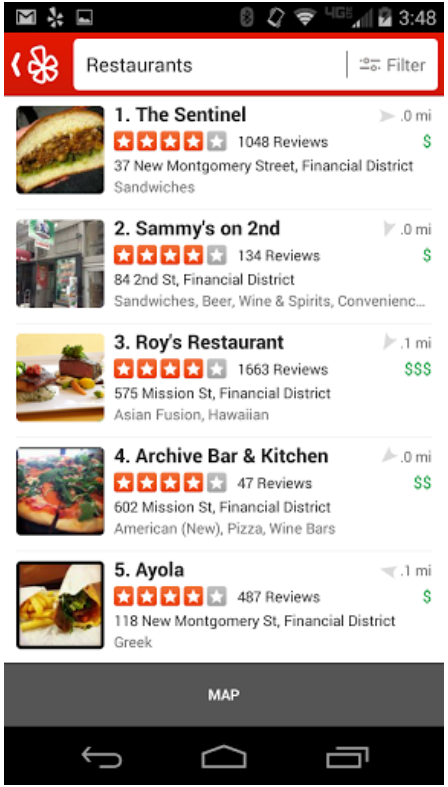
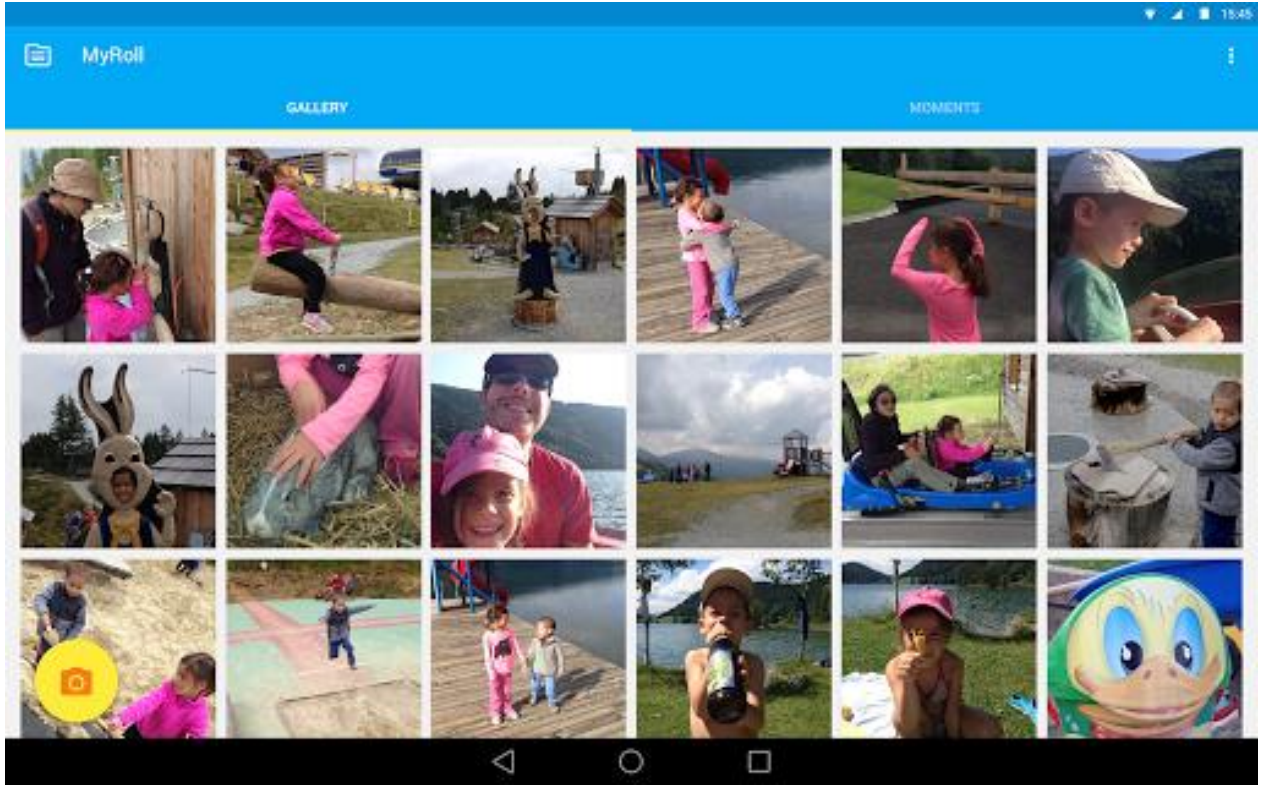
List-based Views

Content partially based on material by Victor Matos

List-based View Examples

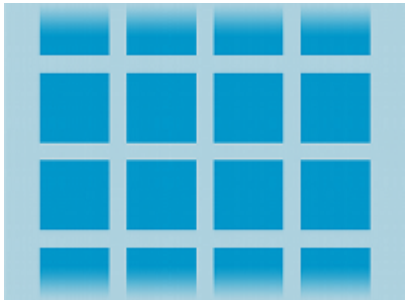




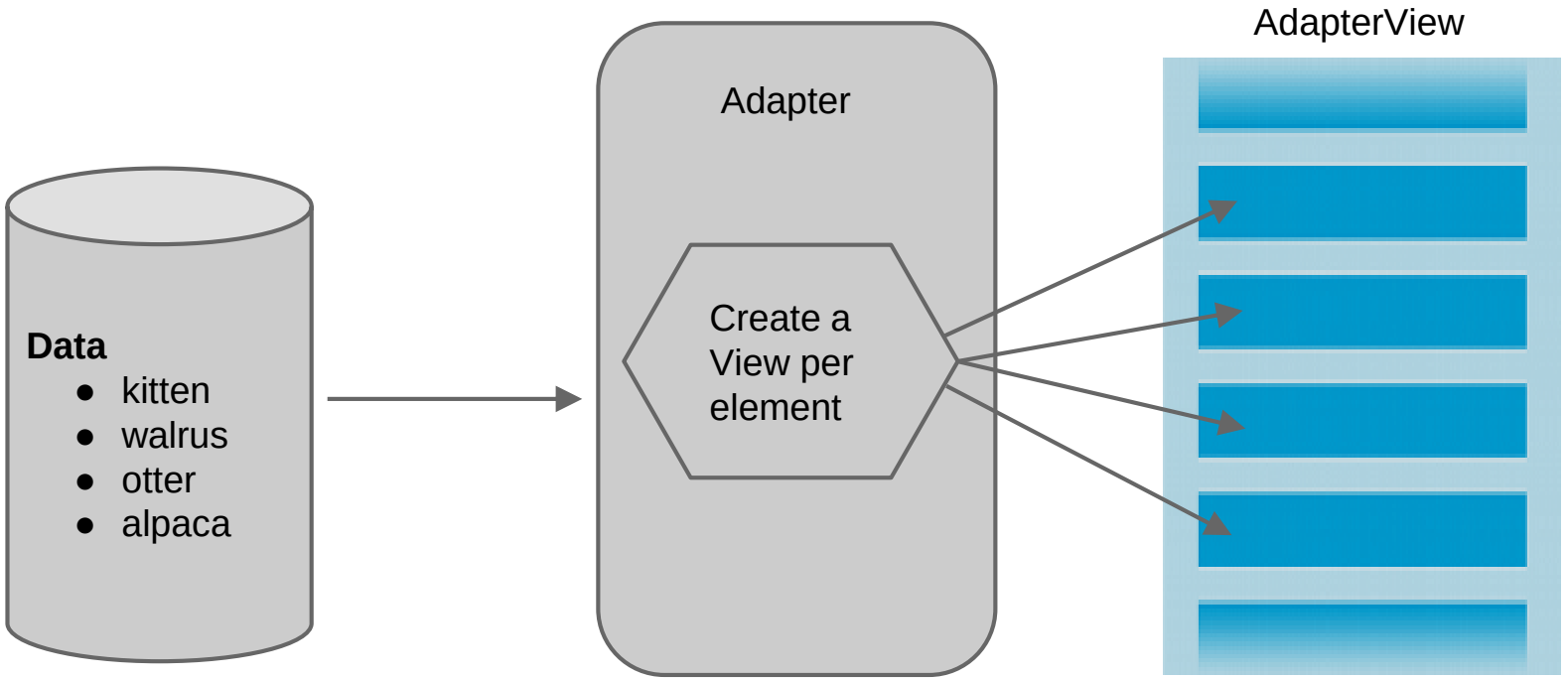
List-based Views

ListView: Displays a single scrolling column list

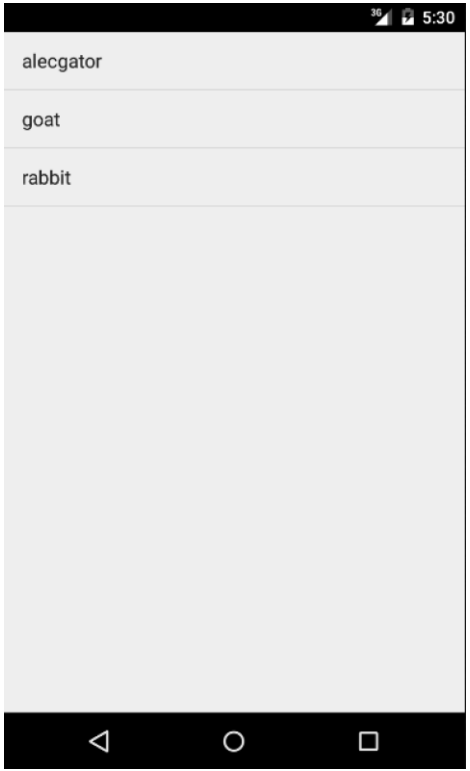
GridView: Displays a scrolling grid of columns and rows.



List-based Views



Simplified ListView Example



Behavior:

- List of words
- Each line is clickable

SimpleListActivity.java

```
public class SimpleListActivity extends ListActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_simple_list);
        String[] items = {"alecgator", "goat", "rabbit"};
        setListAdapter(new ArrayAdapter<String>(this,
                       android.R.layout.simple_list_item_1, items));
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id)
    {
        super.onListItemClick(l, v, position, id);
        Log.d("CS175", "Item clicked: " + position);
    }
}
```

activity_simple.xml

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

    <ListView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@android:id/list"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@android:id/empty"
        android:text="Empty list" />
</RelativeLayout>
```

Summary of Simplified ListView

Activity extends ListActivity

ListActivity class is implicitly bound to element ID

`android:id/list`

Adapter bound to system layout file

`android.R.layout.simple_list_item_1`

Approach only works for limited UI's.

2. Standard ListView Example

Let's use a regular Activity instead of ListActivity.

Now we must explicitly instantiate a ListView object.

ListViewActivity.java

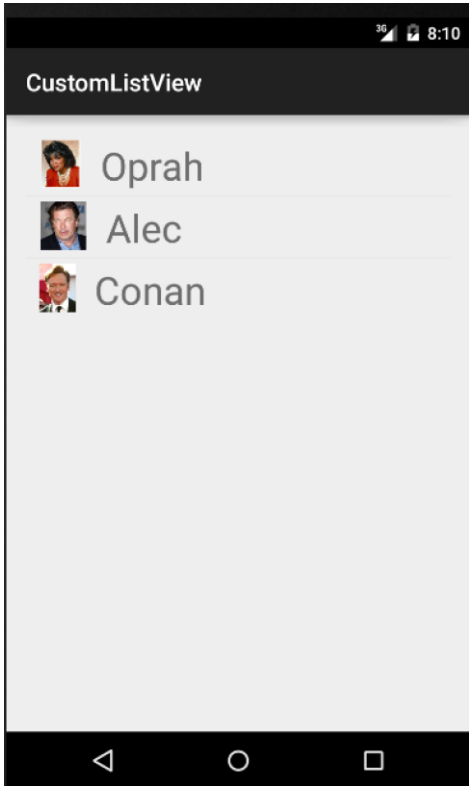
```
public class ListViewActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_view);
        String[] items = {"bat", "cat", "rat"};
        ListView listView = (ListView) findViewById(R.id.listView);
        listView.setAdapter(new ArrayAdapter<String>(this,
                            android.R.layout.simple_list_item_1, items));
        listView.setOnItemClickListener(new
                            AdapterView.OnItemClickListener()
        {
            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                                    int position, long id) {
                Log.d("CS175", "Clicked position: " + position);
            }
        });
    }
}
```

activity_list_view.xml

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

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listView"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

3. Custom ListView



To stylize each row, custom ListViews are used.

custom_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
android:layout_height="match_parent">

    <ImageView
        android:layout_margin="4dp"
        android:layout_width="wrap_content"
        android:layout_height="40dp"
        android:id="@+id/rowImage"/>

    <TextView
        android:layout_margin="4dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Celebrity Name"
        android:textSize="32dp"
        android:id="@+id/rowText" />
</LinearLayout>
```

CustomListViewActivity.java

```
public class CustomListViewActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_custom_list_view);

        ListView listView = (ListView) findViewById(R.id.custom_list_view);
        List<Person> people = new ArrayList<>();
        people.add(new Person("Clooney", "clooney.jpg"));
        people.add(new Person("Rickman", "rickman.jpg"));
        people.add(new Person("Trump", "trump.jpg"));
        listView.setAdapter(new CustomAdapter(this, R.layout.custom_row, people));
    }
}
```

CustomAdapter.java

```
public class CustomAdapter extends ArrayAdapter<Person> {

    private final List<Person> people;
    public CustomAdapter(Context context, int resource, List<Person> people) {
        super(context, resource, people);
        this.people = people;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Person person = people.get(position);
        LayoutInflater inflater = (LayoutInflater) getContext().getSystemService
                (Context.LAYOUT_INFLATER_SERVICE);
        View row = inflater.inflate(R.layout.custom_row, null);

        // Set the text
        TextView textView = (TextView) row.findViewById(R.id.rowText);
        textView.setText(person.getName());

        // Set the image
        try {
            ImageView imageView = (ImageView) row.findViewById(R.id.rowImage);
            InputStream inputStream = getContext().getAssets().open(person.getFilename());
            Drawable drawable = Drawable.createFromStream(inputStream, null);
            imageView.setImageDrawable(drawable);
        } catch (IOException e) { e.printStackTrace();}
        return row;
    }}
```

activity_custom_list_view.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context="edu.csulb.lectures.CustomListViewActivity">
    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/custom_list_view"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

convertView

This parameter is used strictly to increase the performance of your Adapter.

When a ListView uses an Adapter to fill its rows with Views, the adapter populates each list item with a View object by calling getView() on each row. The Adapter uses the convertView as a way of recycling old View objects that are no longer being used.

The ListView can send the Adapter old, "recycled" view objects that are no longer being displayed instead of instantiating an entirely new object each time the Adapter wants to display a new list item.

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = View.inflate(getActivity(), R.layout.item, parent);
    }
    ...
    return convertView;
}
```

ViewHolder

Background: Your code might call findViewById() frequently during the scrolling of ListView, which can slow down performance.

ViewHolder design pattern allows you to avoid repeatedly call findViewById().

A ViewHolder object stores each of the component views inside the tag field of the Layout, so you can immediately access them without the need to look them up repeatedly.

ViewHolder Steps

1. Create a class to hold your exact set of views. For example:

```
static class ViewHolder {
  TextView textView;
  ImageView imageView;
}
```

2. Then populate the ViewHolder and store it inside the layout.

```
ViewHolder holder = new ViewHolder();
holder.imageView = (ImageView)row.findViewById(R.id.rowImage);
holder.textView = (TextView) row.findViewById(R.id.rowText);
row.setTag(holder);
```

3. Access each view without using findViewById.

ViewHolder Example

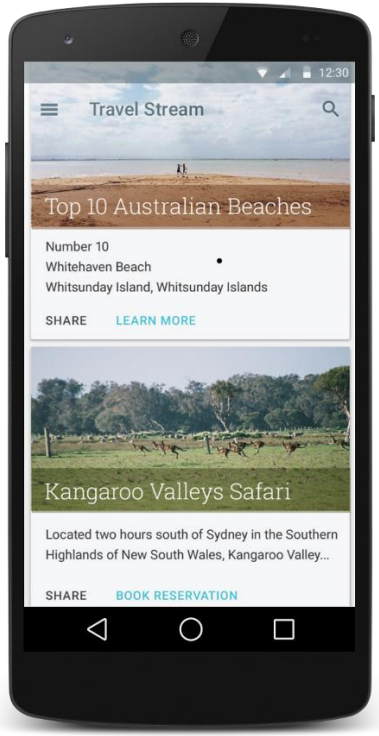
```
public View getView(int position, View convertView, ViewGroup parent)
{   View row = convertView;
    ViewHolder holder = new ViewHolder();
    if (row == null) {
        holder.imageView = (ImageView)row.findViewById(R.id.rowImage);
        holder.textView = (TextView) row.findViewById(R.id.rowText);
        row.setTag(viewHolder);
    } else {
         viewHolder = (ViewHolder)row.getTag();
    }
    viewHolder.textView.setText("Hello");
    ...
    return row;
}
```

CardView

Lets you show information inside cards that have a consistent look across the platform

CardView widgets can have shadows and rounded corners.

CardView extends the FrameLayout class



CardView Example

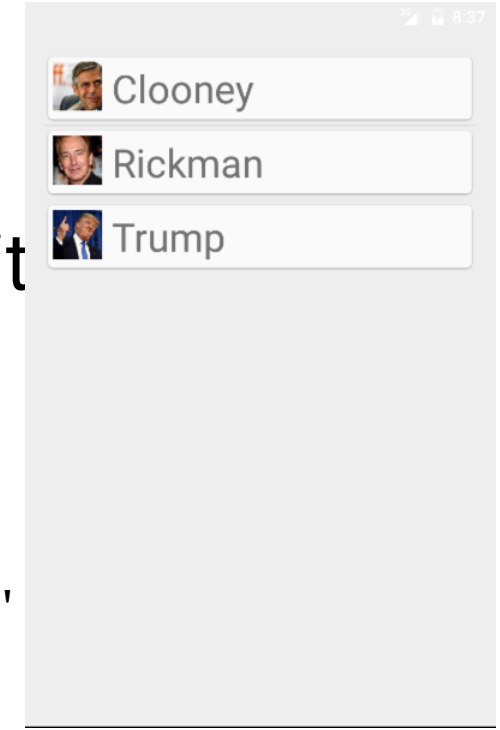
Wrap the item layout in a CardView element

Remove the ListView divider by making it transparent:

```
android:divider="#00000000"
```

Add Gradle dependency:

```
compile 'com.android.support:cardview-v7:21.0.+'
```



CardView XML

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:id="@+id/card_view"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_gravity="center"
    card_view:cardUseCompatPadding="true"
    card_view:cardCornerRadius="4dp">
  <!-- Original Layout -->
</android.support.v7.widget.CardView>
```

RecyclerView

More advanced and flexible version of ListView
Container optimized for displaying large data sets
Tutorial

RecyclerView advantages over ListView

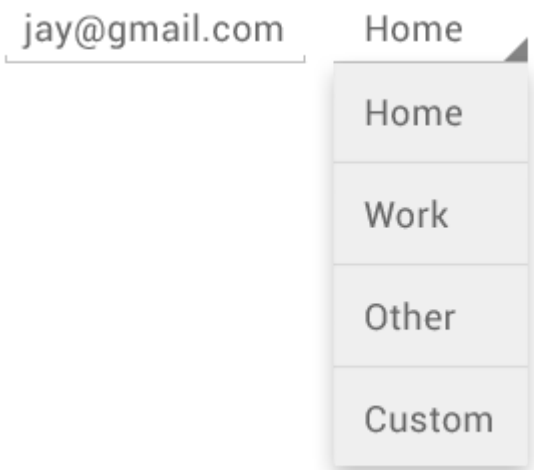
1. Reuse cells while scrolling up/down
   This can also be accomplished with ViewHolders in a ListView, but was optional. In the RecycleView, it's the default way of writing adapter.
2. Decouple the container
   You can put list items easily at run time in the different containers (LinearLayout, GridLayout)
   with setting LayoutManager
   Example:

```
mRecyclerView=(RecyclerView)findViewById(R.id.my_recycler_view);
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
//or
mRecyclerView.setLayoutManager(new GridLayoutManager(this, 2));
```

3. Decouples the animation classes. Animations are delegated to ItemAnimator.

Spinner

- Equivalent to a drop-down
- Same functionality as
  - ListView but less screen space
    - OnItemSelectedListener to listen to callbacks



SpinnerActivity.java

```
public class SpinnerActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_spinner);

        String[] items = {"bat", "cat", "rat"};
        Spinner spinner = (Spinner) findViewById(R.id.spinner);
        spinner.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1,
items));
        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener(){
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                Log.d("CECS453", "Position selected: " + position);
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                Log.d("CS175", "Nothing selected");
            }
        });
    }
}
```

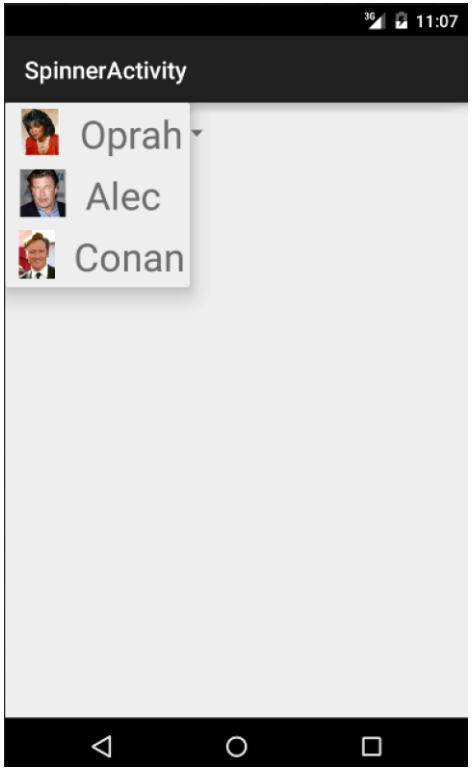
activity_spinner.xml

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

    <Spinner
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/spinner"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

Custom Spinner



We can customize the content of the spinner.

Conceptually the same as customizing the ListView content.

SpinnerActivity.java

```
public class SpinnerActivity extends ActionBarActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_spinner);
        Spinner spinner = (Spinner) findViewById(R.id.spinner);
        // Instead of a List of Strings, use a List of Persons.
        List<Person> person = new ArrayList<>();
        person.add(new Person("Oprah", "oprah.jpeg"));
        person.add(new Person("Alec", "alec.jpeg"));
        person.add(new Person("Conan", "conan.jpeg"));
        spinner.setAdapter(new CustomAdapter(this, R.layout.custom_row, person));

        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                Log.d("CS175", "Position selected: " + position);
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                Log.d("CS175", "Nothing selected");
            }
        });
    }
}
```

activity_spinner.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools=http://schemas.android.com/tools
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <!-- This file remains the same -->
    <Spinner
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/spinner"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

CustomAdapter.java

```
public class CustomAdapter extends ArrayAdapter<Person> {

    private final List<Person> persons;

    public CustomAdapter(Context context, int resource, List<Person> persons) {
        super(context, resource, persons);
        this.persons = persons;
    }

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

    // getDropDownView returns the view for the dropdown. We use the same view
    // between getView and getDropDownView.
    @Override
    public View getDropDownView(int position, View convertView, ViewGroup parent){
        return getView(position);
    }
```

CustomAdapter.java

```
public class CustomAdapter extends ArrayAdapter<Person> {
    ……..
private View getView(int position) {
        LayoutInflater inflater=(LayoutInflater)
                  getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        View row = inflater.inflate(R.layout.custom_row, null);

        TextView textView = (TextView) row.findViewById(R.id.rowText);
        textView.setText(persons.get(position).getName());
        try {
            ImageView imageView = (ImageView) row.findViewById(R.id.rowImage);
            InputStream inputStream =
              getContext().getAssets().open(persons.get(position).getFilename());
            Drawable drawable = Drawable.createFromStream(inputStream, null);
            imageView.setImageDrawable(drawable);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return row;
    }
}
```

AutoCompleteTextView

Specialized version of EditText
Characters typed so far are compared with the beginning of words held in a user-supplied list of suggested values.
The user can choose from the suggestion list or complete typing the word.

CountriesActivity.java

```
public class CountriesActivity extends Activity {
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.countries);
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                        android.R.layout.simple_dropdown_item_1line, COUNTRIES);
        AutoCompleteTextView textView = (AutoCompleteTextView)
                        findViewById(R.id.countries_list);
        textView.setAdapter(adapter);
    }
    private static final String[] COUNTRIES = new String[] {
        "Belgium", "France", "Italy", "Germany", "Spain"
    };
}
```

List-based Views Summary

Examples covered:

- Simple ListView
- Custom ListView
- Spinner
- Custom Spinner
- AutoCompleteTextView

Other examples

GridView - commonly used for images
HorizontalScrollView - horizontal version of ListView