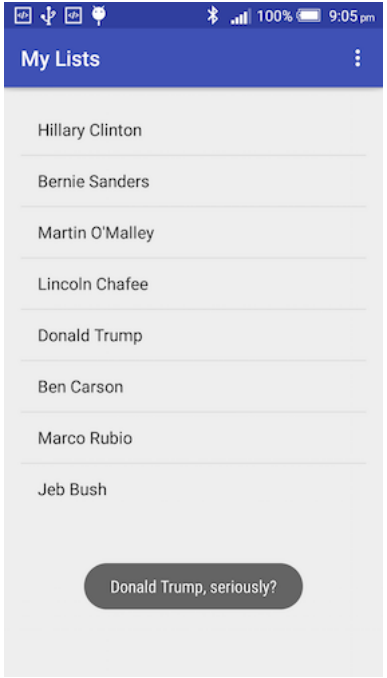
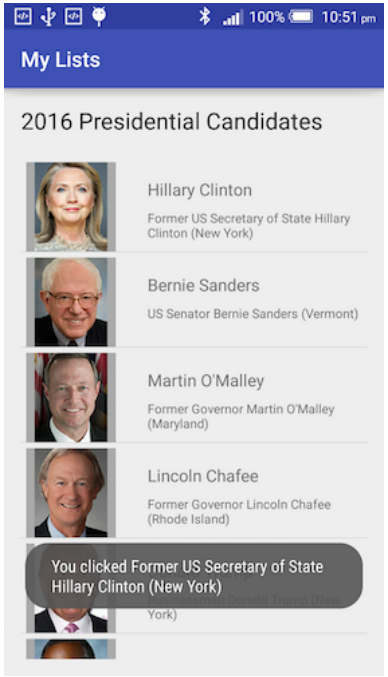
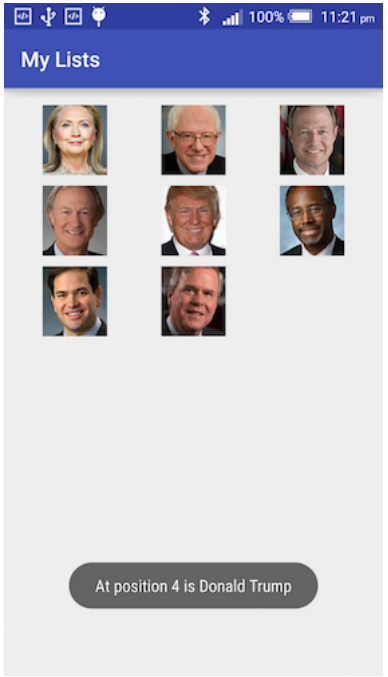
AdapterViews and Fragments

Lab 1

- Simple list
- Complex list
- Grid view



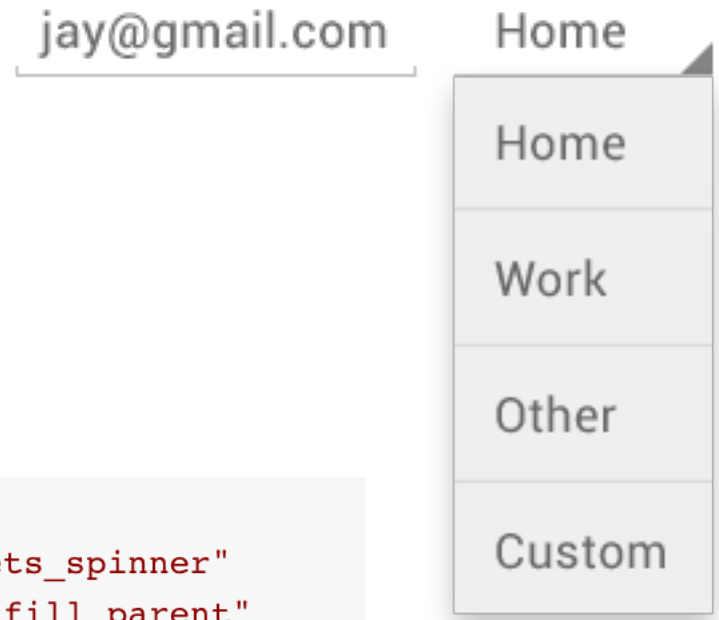




Spinners

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

```xml
<Spinner
    android:id="@+id/planets_spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```



Spinners

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);
// Create an ArrayAdapter using the string array and a default spinner layout
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
        R.array.planets_array, android.R.layout.simple_spinner_item);
// Specify the layout to use when the list of choices appears
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
// Apply the adapter to the spinner
spinner.setAdapter(adapter);
```

Spinners

```
public class SpinnerActivity extends Activity implements OnItemSelectedListener {
    ...

    public void onItemSelected(AdapterView<?> parent, View view,
            int pos, long id) {
        // An item was selected. You can retrieve the selected item using
        // parent.getItemAtPosition(pos)
    }

    public void onNothingSelected(AdapterView<?> parent) {
        // Another interface callback
    }
}
```

http://stackoverflow.com/questions/20151414/how-can-i-use-onitemselected-in-android

Simple list

Data source

```
<string-array name="candidateNames">
    <item>Hillary Clinton</item>
    <item>Bernie Sanders</item>
    <item>Martin O\'Malley</item>
    <item>Lincoln Chafee</item>
    <item>Donald Trump</item>
    <item>Ben Carson</item>
    <item>Marco Rubio</item>
    <item>Jeb Bush</item>
</string-array>

<string-array name="candidateDetails">
    <item>Former US Secretary of State Hillary Clinton (New York)</item>
    <item>US Senator Bernie Sanders (Vermont)</item>
    <item>Former Governor Martin O\'Malley (Maryland)</item>
    <item>Former Governor Lincoln Chafee (Rhode Island)</item>
    <item>Businessman Donald Trump (New York)</item>
    <item>Dr. Ben Carson (Florida)</item>
    <item>US Senator Marco Rubio (Florida)</item>
    <item>Former Governor Jeb Bush (Florida)</item>
</string-array>
```

Adapter view

```
<ListView
    android:id="@+id/listViewSimple"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentTop="true" />
```

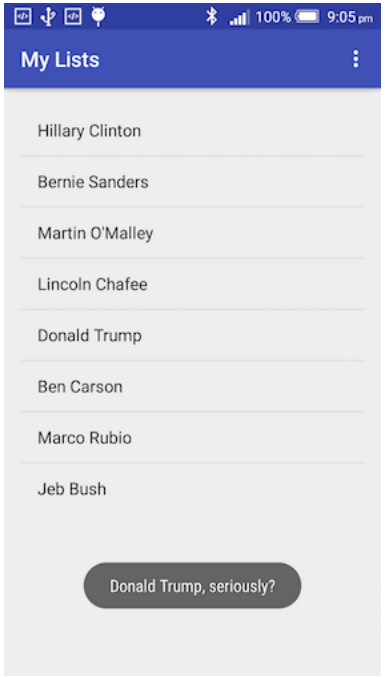
Simple list

```
candidateNames = getResources().getStringArray(R.array.candidateNames);

listView = (ListView) findViewById(R.id.listViewSimple);
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, candidateNa
listView.setAdapter(arrayAdapter);
listView.setOnItemClickListener(

        new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                Toast.makeText(getBaseContext(), candidateNames[position] + ", seriously?", Toast.LENGTH_SHORT).shc
            }
        }
);
```

- `View.getContext()` : Returns the context the view is currently running in. Usually the currently active Activity.
- `Activity.getApplicationContext()` : Returns the context for the entire application (the process all the Activities are running inside of). Use this instead of the current Activity context if you need a context tied to the lifecycle of the entire application, not just the current Activity.
- `ContextWrapper.getBaseContext()` : If you need access to a Context from within another context, you use a ContextWrapper. The Context referred to from inside that ContextWrapper is accessed via getBaseContext().

http://stackoverflow.com/questions/10641144/difference-between-getcontext-getapplicationcontext-getbasecontext-and

simple_list_item_1.xml




```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceListItemSmall"
    android:gravity="center_vertical"
    android:paddingStart="?android:attr/listPreferredItemPaddingStart"
    android:paddingEnd="?android:attr/listPreferredItemPaddingEnd"
    android:minHeight="?android:attr/listPreferredItemHeightSmall" />
```

https://github.com/android/platform_frameworks_base/blob/master/core/res/res/layout/simple_list_item_1.xml

Complex list – data source

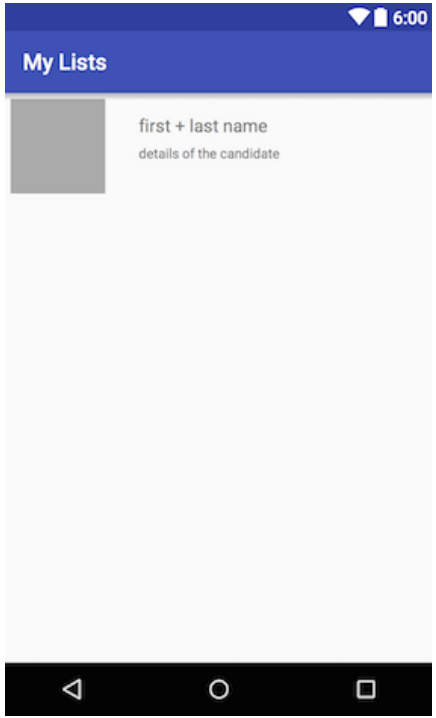
```
private String name;
private String detail;
private int photo;

public Candidate(String name, String detail, int photo) {
    this.name = name;
    this.detail = detail;
    this.photo = photo;
}

public String getName() {
    return name;
}

public String getDetail() {
    return detail;
}

public int getPhoto() {
    return photo;
}

@Override
public String toString() {
    return detail;
}
```

```
candidateNames = getResources().getStringArray(R.array.candidateNames);
candidateDetails = getResources().getStringArray(R.array.candidateDetails);
generateCandidates();
```

```
private void generateCandidates() {

    for (int i = 0; i < candidatePhotos.length; i++) {
        candidates.add(new Candidate(candidateNames[i], candidateDetails[i], candidatePhotos[i]));
    }
}
```

Complex list – layout resource

```
<ImageView
    android:id="@+id/imageView"
    android:layout_width="85dp"
    android:layout_height="85dp"
    android:layout_marginLeft="5dp"
    android:layout_marginTop="5dp"
    android:background="@android:color/darker_gray"
    android:padding="8dp"
    android:scaleType="centerCrop" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginLeft="30dp"
    android:layout_marginTop="20dp"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textViewName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="first + last name"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/textViewDetail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="details of the candidate"
        android:textSize="12sp" />

</LinearLayout>
```



Complex list - AdapterView

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:text="2016 Presidential Candidates"
    android:textAppearance="?android:attr/textAppearanceLarge" />

<ListView
    android:id="@+id/listViewComplex"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView"
    android:layout_marginTop="20dp" />
```

Complex list - Adapter

```
listView.setAdapter(new CandidateAdapter(this, R.layout.list_item, candidates));
```

```
public class CandidateAdapter extends ArrayAdapter<Candidate> {

    private int resource;
    private ArrayList<Candidate> candidates;
    private Context context;

    public CandidateAdapter(Context context, int resource, ArrayList<Candidate> candidates) {
        super(context, resource, candidates);
        this.resource = resource;
        this.candidates = candidates;
        this.context = context;
    }
```

Complex list - Adapter

```
@NonNull
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View v = convertView;
    try {
        if (v == null) {
            LayoutInflater layoutInflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVI
            v = layoutInflater.inflate(resource, parent, false);
        }

        ImageView imageView = (ImageView) v.findViewById(R.id.imageView);
        TextView textViewName = (TextView) v.findViewById(R.id.textViewName);
        TextView textViewDetail = (TextView) v.findViewById(R.id.textViewDetail);

        imageView.setImageResource(candidates.get(position).getPhoto());
        textViewName.setText(candidates.get(position).getName());
        textViewDetail.setText(candidates.get(position).getDetail());

    } catch (Exception e) {
        e.printStackTrace();
        e.getCause();
    }
    return v;
```

GridView

Developers | Design | Develop | Distribute

Training | API Guides | Reference | Tools | Google Services | Samples

- Introduction
- App Components
- App Resources
- App Manifest
- User Interface
  - Overview
  - Layouts
    - Linear Layout

Grid View

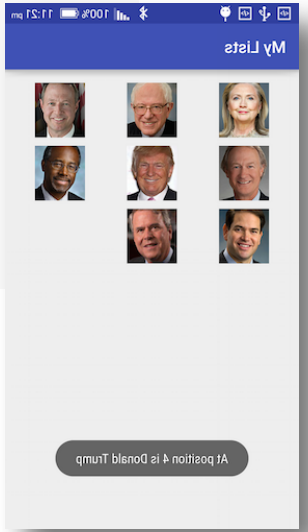
`GridView` is a `ViewGroup` that displays items in a two-dimensional, scrollable grid. The grid items are automatically inserted to the layout using a `ListAdapter`.

For an introduction to how you can dynamically insert views using an adapter, read Building Layouts with an Adapter.

Grid view

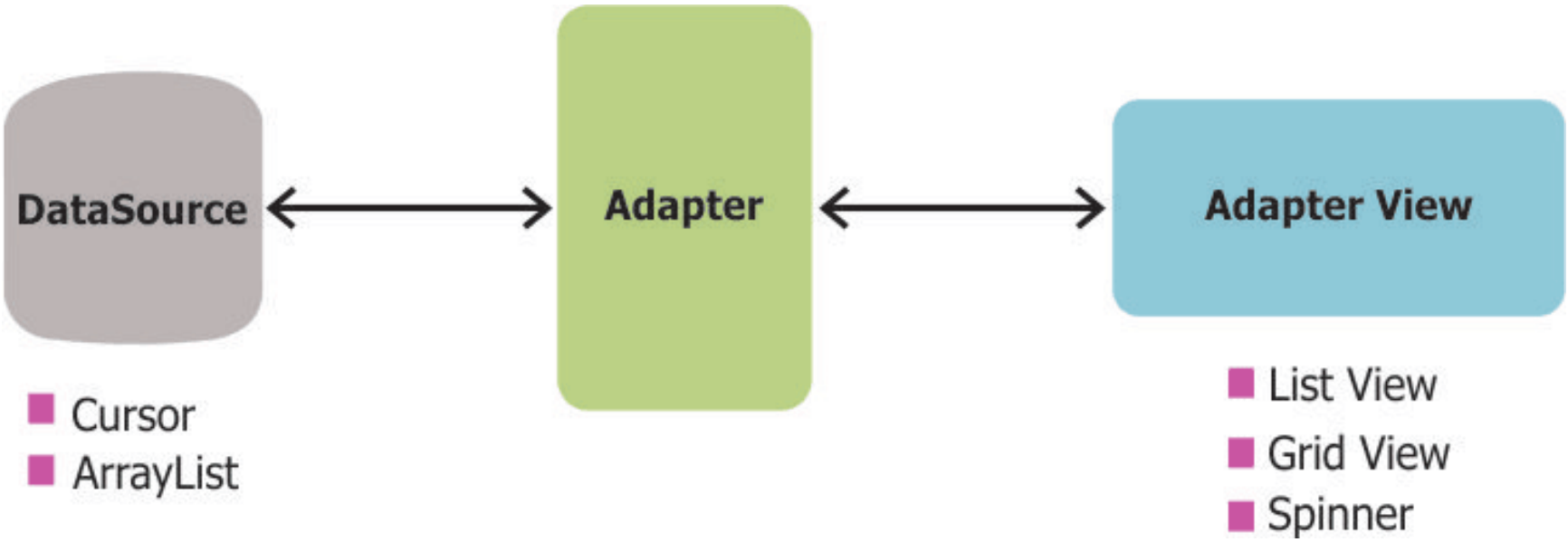
```
android:columnWidth="90dp"
android:gravity="center"
android:horizontalSpacing="10dp"
android:numColumns="auto_fit"
android:stretchMode="columnWidth"
android:verticalSpacing="10dp"
```

```
public View getView(int position, View convertView, ViewGroup parent) {
    ImageView imageView;
    if (convertView == null) {
        imageView = new ImageView(context);
        imageView.setLayoutParams(new GridView.LayoutParams(200, 200));
        imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
        imageView.setPadding(8, 8, 8, 8);
    } else {
        imageView = (ImageView) convertView;
    }

    imageView.setImageResource(candidatePhotos[position]);
    return imageView;
}
```




```
public class ImageAdapter extends BaseAdapter {

    private int[] candidatePhotos = PhotoListActivity.candidatePhotos;
    private Context context;
```

AdapterView



Adapter

Building Layouts with an Adapter

When the content for your layout is dynamic or not pre-determined, you can use a layout that subclasses `AdapterView` to populate the layout with views at runtime. A subclass of the `AdapterView` class uses an `Adapter` to bind data to its layout. The `Adapter` behaves as a middleman between the data source and the `AdapterView` layout—the `Adapter` retrieves the data (from a source such as an array or a database query) and converts each entry into a view that can be added into the `AdapterView` layout.

`ArrayAdapter`

`CursorAdapter`

`SimpleCursorAdapter`

AdapterView

AdapterView

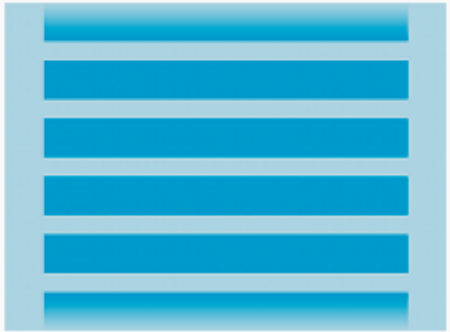
extends ViewGroup

java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.AdapterView<T extends android.widget.Adapter>

▸ Known Direct Subclasses
AbsListView, AbsSpinner, AdapterViewAnimator

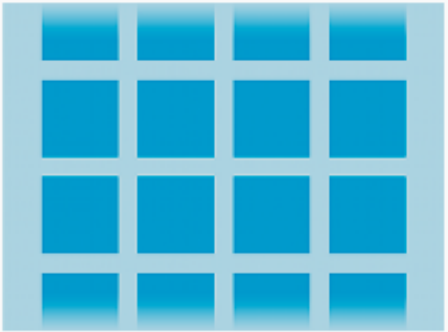
▸ Known Indirect Subclasses
AdapterViewFlipper, ExpandableListView, Gallery, GridView, ListView, Spinner, StackView

List View



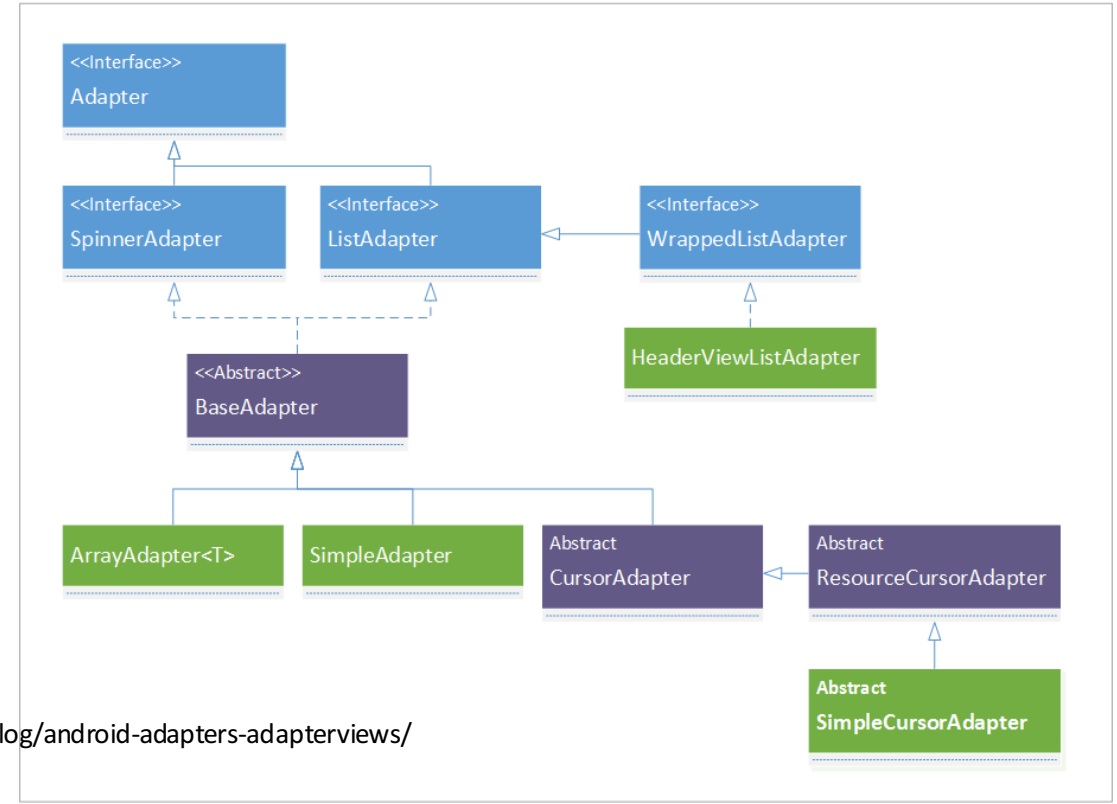
Displays a scrolling single column list.

Grid View



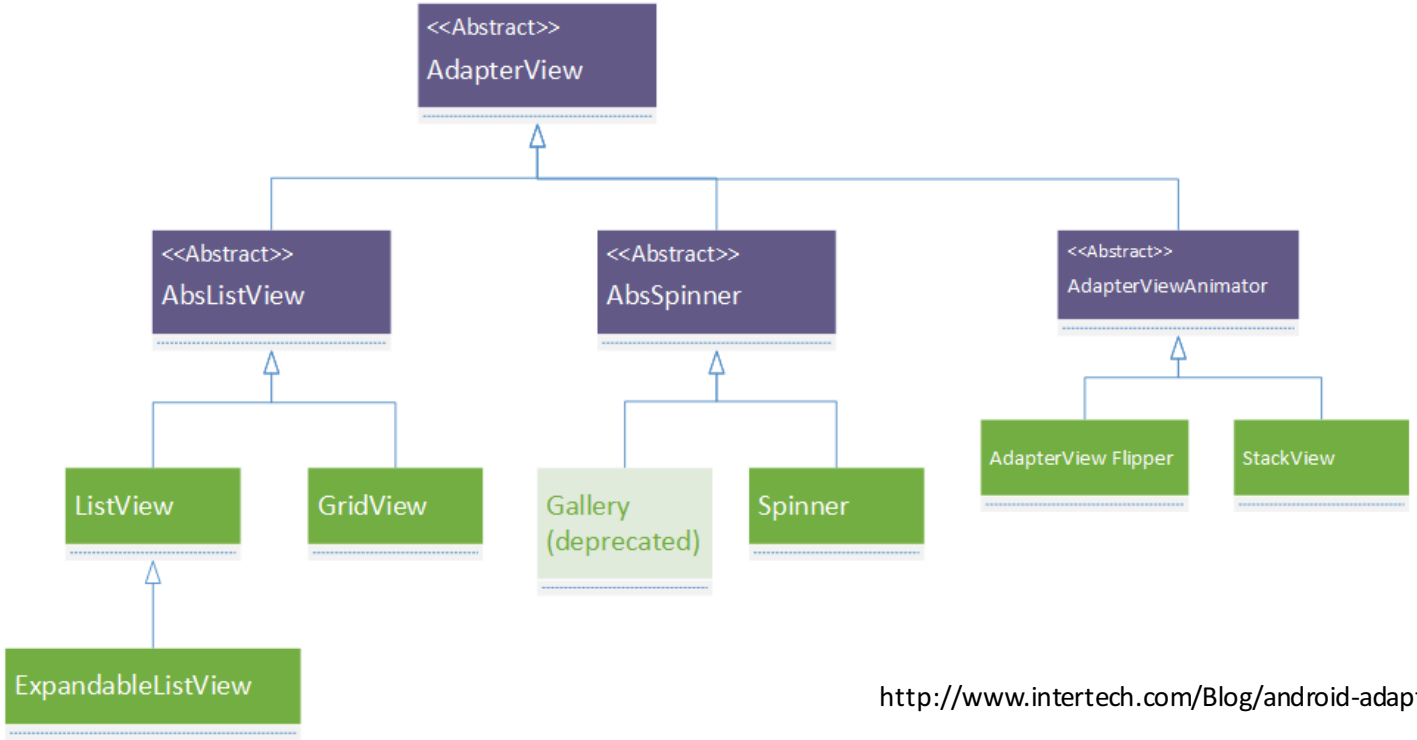
Displays a scrolling grid of columns and rows.

Hierarchy of Adapters



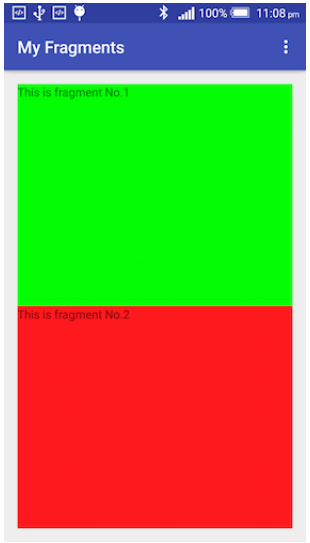
http://www.intertech.com/Blog/android-adapters-adapterviews/

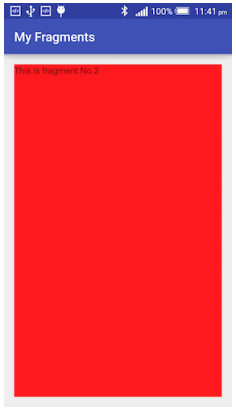
AdapterView hierarchy



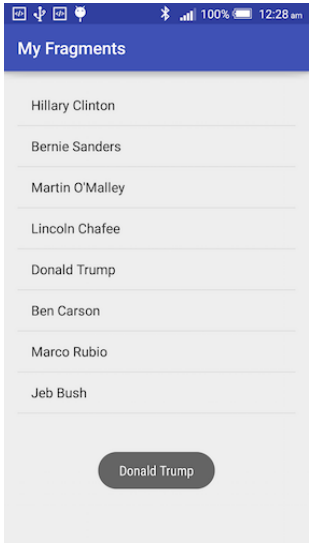
http://www.intertech.com/Blog/android-adapters-adapterviews/

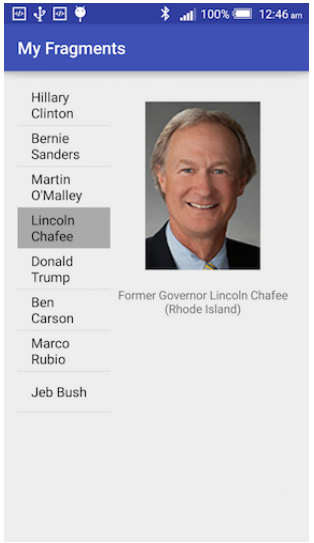
Lab 2


100% 11:08 pm
My Fragments
This is fragment No.1
This is fragment No.2


100% 11:41 pm
My Fragments
This is fragment No.2


100% 11:42 pm
My Fragments
This is fragment No.1

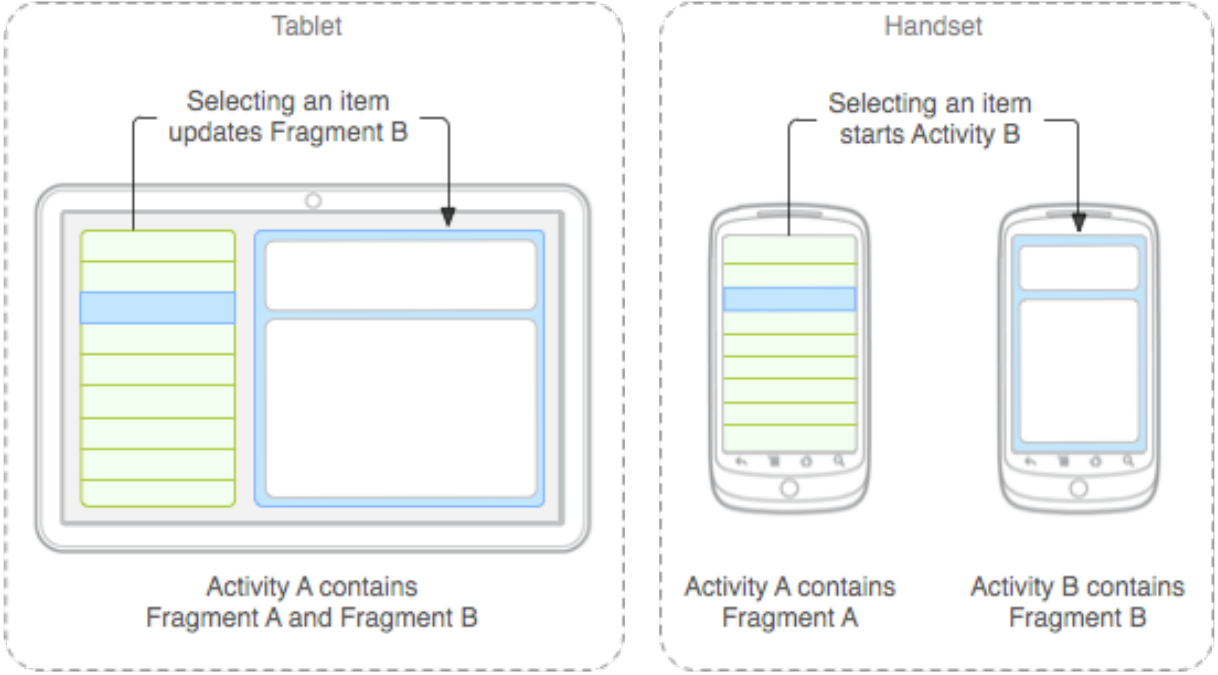

100% 12:28 am
My Fragments
Hillary Clinton
Bernie Sanders
Martin O'Malley
Lincoln Chafee
Donald Trump
Ben Carson
Marco Rubio
Jeb Bush
Donald Trump


100% 12:46 am
My Fragments
Hillary Clinton
Bernie Sanders
Martin O'Malley
Lincoln Chafee
Donald Trump
Ben Carson
Marco Rubio
Jeb Bush
Former Governor Lincoln Chafee (Rhode Island)

Lab 2

- Introducing fragments
- Static fragments
- Adding fragments dynamically
- Passing data between fragments
- Fragment lifecycle

Introducing fragments



Introducing fragments

- Views — Views are the base class for all visual interface elements (commonly known as controls or widgets).
- View Groups — View Groups are extensions of the View class that can contain multiple child Views.
- Fragments — Fragments, introduced in Android 3.0 (API level 11), are used to encapsulate portions of your UI.
- Activities — represent the window, or screen, being displayed.

Introducing fragments

- DialogFragment
  - Using this class to create a dialog is a good alternative to using the dialog helper methods in the Activity class
- ListFragment
  - It provides several methods for managing a list view, such as the onListItemClick() callback to handle click events.
- PreferenceFragment
  - Displays a hierarchy of Preference objects as a list, similar to PreferenceActivity. This is useful when creating a "settings" activity for your application.

Static fragments

Step 1: fragment layout 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="#00FF00"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="This is fragment #1"
    android:textColor="#000000"
    android:textSize="25sp" />
</LinearLayout>
```

Static fragments

Step 2: extend Fragment class

```
public static class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.example_fragment, container, false);
    }
}
```

Static fragments

Step 2: extend Fragment class

- The **resource ID** of the layout you want to inflate.
- The **ViewGroup** to be the parent of the inflated layout. Passing the container is important in order for the system to apply layout parameters to the root view of the inflated layout, specified by the parent view in which it's going.
- A **boolean** indicating whether the inflated layout should be attached to the ViewGroup (the second parameter) during inflation.

Static fragments

Step 3: add fragment layout into activity layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ArticleListFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ArticleReaderFragment"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

Adding fragments dynamically

```
FragmentManager fragmentManager = getFragmentManager()
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

```
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

Adding fragments dynamically

FragmentManger

- Get fragments that exist in the activity, with findFragmentById() (for fragments that provide a UI in the activity layout) or findFragmentByTag() (for fragments that do or don't provide a UI).
- Pop fragments off the back stack, with popBackStack() (simulating a Back command by the user).
- Register a listener for changes to the back stack, with addOnBackStackChangedListener().

Adding fragments dynamically

```
WindowManager aWindowManager = getWindowManager();
Display aDisplay = aWindowManager.getDefaultDisplay();

int orientation = aDisplay.getRotation();

if (orientation == Surface.ROTATION_90 || orientation == Surface.ROTATION_270) {
    BlankFragment frag1 = new BlankFragment();
    aFragmentTransaction.replace(R.id.container, frag1);
} else {
```

Windows: left Ctrl + F12

Mac: Fn + Ctrl + F12

FragmentTransaction:

- add
- remove
- replace
- commit
- addToBackStack

stackoverflow

Questions | Tags | Users | Badge

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

Android Fragments: When to use hide/show or add/remove/replace?

38

13

Suppose I wish to replace the current fragment in some container view with another. Is it better to use replace...

```
FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
ft.replace(R.id.fragment_container, newFragment, null);
ft.commit();
```

... or the following, with show and hide?

http://stackoverflow.com/questions/13149446/android-fragments-when-to-use-hide-show-or-add-remove-replace

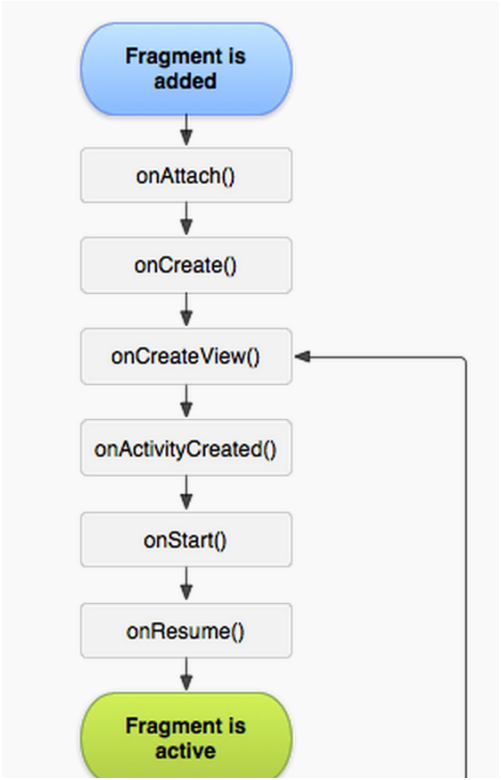
Passing data between fragments

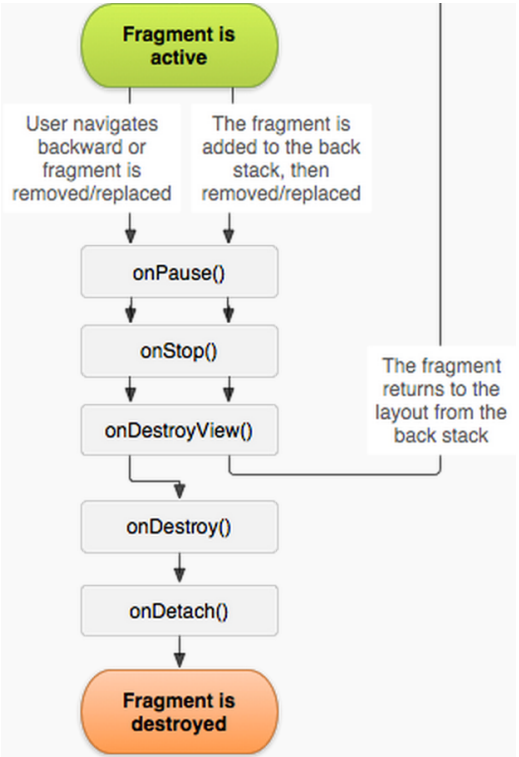
```
View listView = getActivity().findViewById(R.id.list);
```

```
ExampleFragment fragment =
        (ExampleFragment) getFragmentManager().findFragmentById(R.id.example_fragment);
```

Fragment lifecycle

- Resumed
- Paused
- Stopped





Fragment lifecycle

`onAttach()`
: Called when the fragment has been associated with the activity (the `Activity` is passed in here).

`onCreateView()`
: Called to create the view hierarchy associated with the fragment.

`onActivityCreated()`
: Called when the activity's `onCreate()` method has returned.

`onDestroyView()`
: Called when the view hierarchy associated with the fragment is being removed.

`onDetach()`
: Called when the fragment is being disassociated from the activity.