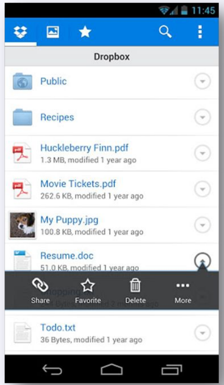
CS196

Android

ListViews and Adapters

Recap

- A view that shows items in a vertically scrolling list
- The items come from the ListAdapter associated with this view



11:45

Dropbox

Public

Recipes

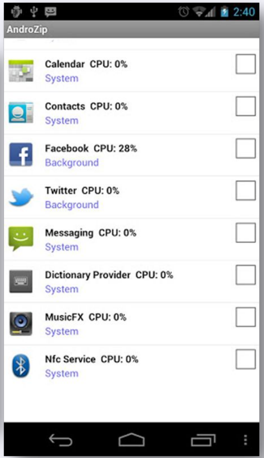
Huckleberry Finn.pdf
1.3 MB, modified 1 year ago

Movie Tickets.pdf
262.6 KB, modified 1 year ago

My Puppy.jpg
100.8 KB, modified 1 year ago

Resume.doc
51.0 KB, modified 1 year ago

Todo.txt
36 Bytes, modified 1 year ago



2:40

AndroZip

Calendar CPU: 0%
System

Contacts CPU: 0%
System

Facebook CPU: 28%
Background

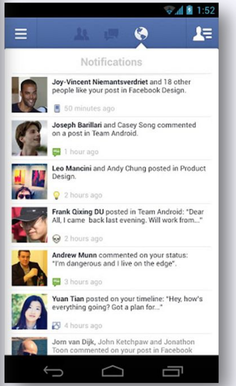
Twitter CPU: 0%
Background

Messaging CPU: 0%
System

Dictionary Provider CPU: 0%
System

MusicFX CPU: 0%
System

Nfc Service CPU: 0%
System



1:52

Notifications

Joy-Vincent Niemantsverdriet and 18 other people like your post in Facebook Design.
50 minutes ago

Joseph Barillari and Casey Song commented on a post in Team Android.
1 hour ago

Leo Mancini and Andy Chung posted in Product Design.
2 hours ago

Frank Qixing DU posted in Team Android: "Dear All, I came back last evening. Will work from..."
2 hours ago

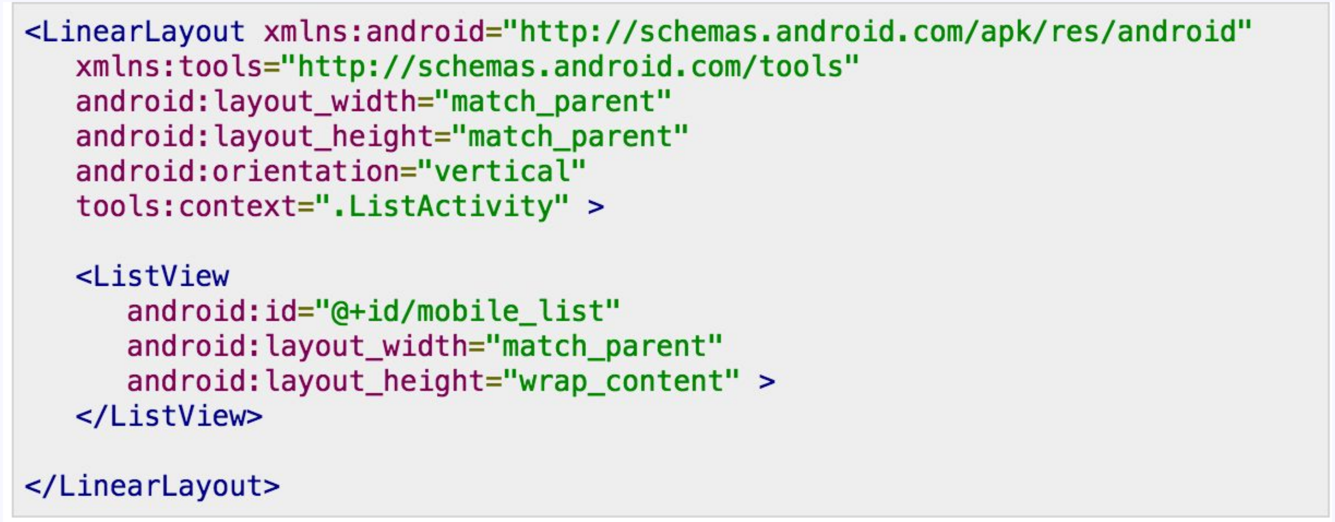
Andrew Munn commented on your status: "I'm dangerous and I live on the edge".
3 hours ago

Yuan Tian posted on your timeline: "Hey, how's everything going? Got a plan for..."
4 hours ago

Jorn van Dijk, John Ketchpaw and Jonathon Toon commented on your post in Facebook

Setting up the XML

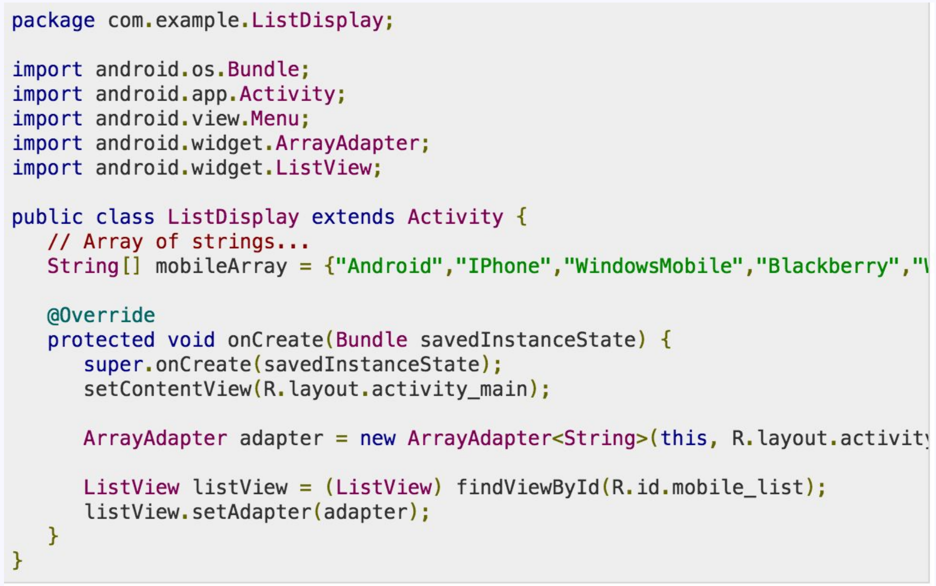
- Declare a ListView element in your layout
- Declare an ID, width, and height



```xml
<LinearLayout 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:orientation="vertical"
   tools:context=".ListActivity" >

   <ListView
      android:id="@+id/mobile_list"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" >
   </ListView>

</LinearLayout>
```

Java Setup

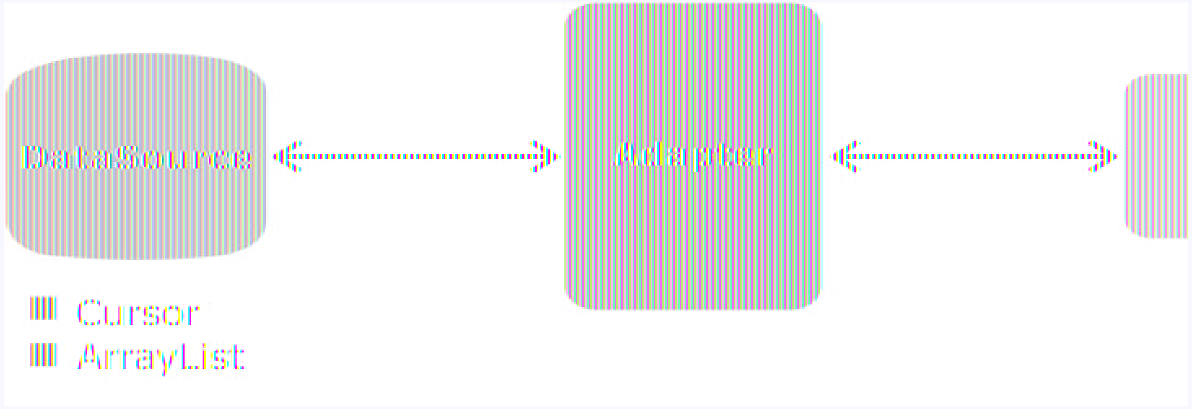
- Declare a ListView object
- Use findViewById to tell the app what listview you're referring to



```
package com.example.ListDisplay;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class ListDisplay extends Activity {
   // Array of strings...
   String[] mobileArray = {"Android","IPhone","WindowsMobile","Blackberry","\

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

      ArrayAdapter adapter = new ArrayAdapter<String>(this, R.layout.activit

      ListView listView = (ListView) findViewById(R.id.mobile_list);
      listView.setAdapter(adapter);
   }
}
```

Adapters

- Something that converts data into the proper representation in the UI
- Data -> Adapter -> Screen
- Many different types of adapters (default, custom)



ArrayAdapters

- Simplest type of adapter
- Takes an array of Java objects and utilizes the toString() method to map the data to a single row within the listview
- Maps a string to a textview
- Will not work with custom objects

CursorAdapter

- Takes a cursor and adapts it to a listview
- A cursor is a set of data from a database query
- Acts very similar to an ArrayAdapter

CustomAdapter

- Used for displaying data in a custom representation
- Create a custom class that extends the ArrayAdapter class
- Implement a constructor and override the getView() function
- Create a custom xml layout file that represents how the data will be displayed

getView()

- Gets a view that displays data at the specified position in the data set
- Parameters: position, convertView, parent
  - Position: int that represents the position of the item within the adapter’s data set
  - ConvertView: The old view to reuse if possible. Otherwise this is set to null
  - Parent: ViewGroup that the view will eventually attach to

Using the Adapter

- Declare an instance of your adapter in your activity/fragment
- Call the setAdapter() function on your listview and pass in the adapter
- Add data to the adapter by passing in a single item or whole collection

```
// Add item to adapter
User newUser = new User("Nathan", "San Diego");
adapter.add(newUser);
// Or even append an entire new collection
// Fetching some data, data has now returned
// If data was JSON, convert to ArrayList of User objects.
JSONArray jsonArray = ...;
ArrayList<User> newUsers = User.fromJson(jsonArray)
adapter.addAll(newUsers);
```

ViewHolders

- ViewHolders improve the performance of the data population
- Caches view lookups, reduces calls to findViewById()
- Loads item faster

Implement the ViewHolder Pattern

- Declare a static class within the custom adapter class called “ViewHolder”
- Change the implementation within the getView() function

```java
public class UsersAdapter extends ArrayAdapter<User> {
    // View lookup cache
    private static class ViewHolder {
        TextView name;
        TextView home;
    }

    public UsersAdapter(Context context, ArrayList<User> users) {
       super(context, R.layout.item_user, users);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
       // Get the data item for this position
       User user = getItem(position);
       // Check if an existing view is being reused, otherwise inflate the view
       ViewHolder viewHolder; // view lookup cache stored in tag
       if (convertView == null) {
          // If there's no view to re-use, inflate a brand new view for row
          viewHolder = new ViewHolder();
          LayoutInflater inflater = LayoutInflater.from(getContext());
          convertView = inflater.inflate(R.layout.item_user, parent, false);
          viewHolder.name = (TextView) convertView.findViewById(R.id.tvName);
          viewHolder.home = (TextView) convertView.findViewById(R.id.tvHome);
          // Cache the viewHolder object inside the fresh view
          convertView.setTag(viewHolder);
       } else {
           // View is being recycled, retrieve the viewHolder object from tag
           viewHolder = (ViewHolder) convertView.getTag();
       }
       // Populate the data into the template view using the data object
       viewHolder.name.setText(user.name);
       viewHolder.home.setText(user.hometown);
       // Return the completed view to render on screen
       return convertView;
   }
}
```

Useful Link

https://github.com/codepath/android_guides/wiki/Using-an-ArrayAdapter-with-ListView