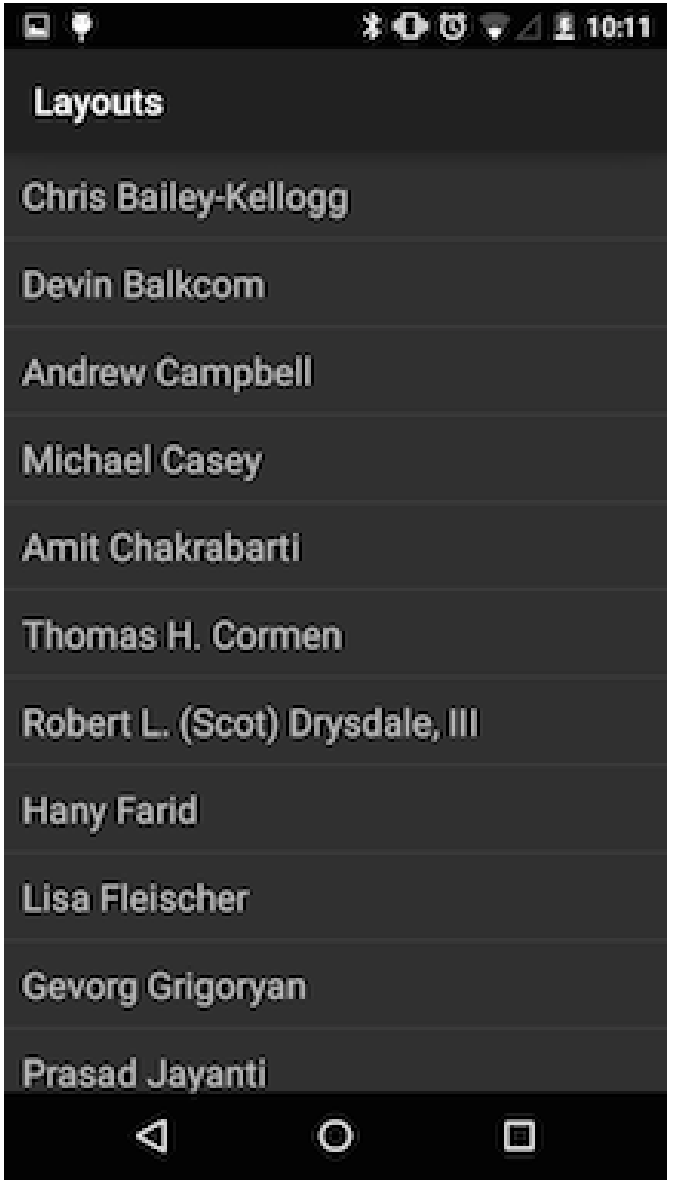
IT – Entrepreneurship
User Interface - 2

ListView Layout

- If you want to design a UI with a long list of items then the list view is for you.
- The ListView controller allows you to vertically scrawl through a list of items



ListView Layout

- A list view is fundamentally created using 3 basic components:
- Listview layout
- Item holder layout
- Adapter

ListView Layout

- The ListView layout is the container where the list is populated
- Holder layout is the xml provided to each element of the list
- Adapter is used to provide control and to bind data and views of the list.

Adapter

- Android uses Adapters to provide the data to the ListView object
- The adapter also defines how each row is the ListView is displayed
- We first get the listview set up
- Then set up the setOnItemClickListener for the list view for each of the entries in the array
- The ArrayAdapter object is used to bind the data to the ListView prior to setListAdapter displaying the view to the screen.

Item OnClick

- Set up the listView.setOnItemClickListener
- When the user clicks and item the onItemClick callback executes.
- onItemClick() callback method is when an item in the AdapterView has been clicked by the user

Item OnClick

- The parameters provided by the callback include:
- parent: the AdapterView where the click happened
- view: the view within the AdapterView that was clicked
- position: the position of the view in the adapter
- id: the row id of the item that was clicked

Final Points

- A ListView object can be bound to different data sources.
- Adapters allow you to bind the data source to the view.
- It serves as an intermediary between the data and view.
- An AdapterView is a view whose children are determined by an Adapter e.g., ListView

Final Points

- 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.
- ListView is a subclass of the AdapterView class and uses an Adapter to bind data to its layout.

Other layouts
using date and time pickers

Date and Time pickers

- Android provides a set of standard widgets for setting the date and time, these are called Pickers
- The DatePickerDialog object is first created using the constructor and then displayed to the user using the show() method

Date Picker

- Create a listener to respond to the results from date picker

```
DatePickerDialog.OnDateSetListener mDateListener = new
DatePickerDialog.OnDateSetListener() {
                        public void onDateSet(DatePicker view, int year, int monthOfYear,
                                        int dayOfMonth) {
                                mDateAndTime.set(Calendar.YEAR, year);
                                mDateAndTime.set(Calendar.MONTH, monthOfYear);
                                mDateAndTime.set(Calendar.DAY_OF_MONTH, dayOfMonth);
                                updateDateAndTimeDisplay();
                        }
                };
```

Date Picker

- Then pass the listener to DatePicker object along with the current date setting

```
new DatePickerDialog(DateAndTimeActivity.this, mDateListener,
                                    mDateAndTime.get(Calendar.YEAR),
                                    mDateAndTime.get(Calendar.MONTH),
                                    mDateAndTime.get(Calendar.DAY_OF_MONTH)).show();
```

Time Picker

```
TimePickerDialog.OnTimeSetListener mTimeListener = new
TimePickerDialog.OnTimeSetListener() {
                        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
                                mDateAndTime.set(Calendar.HOUR_OF_DAY, hourOfDay);
                                mDateAndTime.set(Calendar.MINUTE, minute);
                                updateDateAndTimeDisplay();
                        }
                };
```

Time Picker

```
new TimePickerDialog(DateAndTimeActivity.this, mTimeListener,
                                    mDateAndTime.get(Calendar.HOUR_OF_DAY),
                                    mDateAndTime.get(Calendar.MINUTE), true).show();
```

ScrollView

- Many time you will want to design a layout that has too many views
- Android has a scrollable view that allows you to load up the layout
- The user simply swipes down and up to get to the view of interest

SharedPrefences

- You can use a simple SharedPreference object to store small amounts of user data.
- For more sophisticated data storage we will use databases and particular SQLite
- A SharedPreferences object points to a file containing key-value pairs and provides simple methods to read and write them.
- Each SharedPreferences file is managed by the framework and can be private or shared.

Agenda

- Get a Handle to a SharedPreferences
- Write to Shared Preferences
- Read from Shared Preferences

Get a Handle to a SharedPreferences

You can create a new shared preference file or access an existing one by calling one of two methods

- getSharedPreferences() — Use this if you need multiple shared preference files identified by name, which you specify with the first parameter.

  You can call this from any Context in your app.

Get a Handle to a SharedPreferences

- getPreferences() — Use this from an Activity if you need to use only one shared preference file for the activity. Because this retrieves a default shared preference file that belongs to the activity, you don't need to supply a name.

Get a Handle to a SharedPreferences

- The following code accesses the shared preferences file that's identified by the resource string R.string.preference_file_key and opens it using the private mode so the file is accessible by only your app.

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences(
        getString(R.string.preference_file_key), Context.MODE_PRIVATE);
```

Get a Handle to a SharedPreferences

- Alternatively, if you need just one shared preference file for your activity, you can use the getPreferences() method

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

Write to Shared Preferences

- To write to a shared preferences file, create a SharedPreferences.Editor by calling edit() on your SharedPreferences.
- Pass the keys and values you want to write with methods such as putInt() and putString().
- Then call commit() to save the changes.

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putInt(getString(R.string.saved_high_score), newHighScore);
editor.commit();
```

Read from Shared Preferences

- To retrieve values from a shared preferences file, call methods such as getInt() and getString()
- providing the key for the value you want, and optionally a default value to return if the key isn't present.

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.string.saved_high_score_default);
long highScore = sharedPref.getInt(getString(R.string.saved_high_score), defaultValue);
```

References

- cs.dartmouth.edu
- Android Developers