EECS4443 Notes

Jerry Wu

2023-09-18

Layouts

XML advantages

- XML is used exclusively for the view. We can separate concerns between function and form by using MVC.
- The ID attribute: `android:id="@+id/decbutton"` signifies that we are pulling a value, in this case, a button from the `R` (resources) class.
- `R.java` is automatically generated based on what is in the XML view. This allows us to change a single value in the XML to change **all** occurences of that value in code.

Layout parameters

An app activity can have multiple layouts.

- Layouts take on the form of `layout_foo`
- `LinearLayout`: every element is stacked sequentially on the screen depending on the orientation of the screen.

What is a layout?

Simply put, a layout is what defines the UI elements in a given activity (screen). It is the **parent view** to all other elements within it. A `Layout` is a `ViewGroup`, and a `ViewGroup` is a `View`

Dynamic layout

In a dynamic layout, the elements on the screen are not pre determined. In other words, `AdapterView` is used to populate the layout with views **at runtime** like `ListView`, `GridView`, etc. These are very common views for organizing data in many applications, since they display items in either a list or a grid respectively.

- When scrolling through a list of elements, we don't want to have them all rendered at once, as that will cause performance issues (low memory). We can deal with these issues by using dynamic rendering. To achieve this, we only render what is on the screen at the moment (text, images, etc.) when they are scrolled to.

Tabs

- Tabs allow us to create muti paged activities
- All items are displayed concurrently
- Tabs should all be equal width with a text label
- Tabs are scrollable horizontally to find more pages within a single activity. In other words, we can switch between views without switching activities.
- They are NOT like tabs in a browser, because a tab in a browser can be considered as another activity, if nothing else.

Spinners

- It's like a `ListView`, but it is in the form of a dropdown menu.

Buttons

- We shouldn't use buttons too often since they have a heavy appearance. They are most suitable for important actions like sign up/log in.
- Rule of thumb is to not have more than 2 buttons on the screen at any given time.

Image buttons > text buttons

- Image buttons are good for representing actions with easy to understand indicators such as a magnifying glass for search, etc.
- Image buttons save a lot of space.
- We can combine images and text in a button in order to add more context as to what the button does.
- Coloured image buttons are not necessary since users are used to interacting with objects. In fact, it is discouraged since it makes applications look archaic.
- Same case applies to text buttons, however, the text colour should be different from the body text of the application so as to avoid roach motel (disguised links).

SeekBar, slider, and progress indicator

- Seekbar is good for increasing and decreasing continuous/discrete values like volume, brightness, timestamps in audio/video, etc. Though the former two are sliders.
- Progress indicators indicate how far a task has progressed such as download-/install progress. Can only increase; fill from 0% to 100%.
- If we don't know exactly when the task will complete, we can use an **activity indicator** (spinning loading circle) to avoid giving the user false despair.

Switches

- **Checkbox** - used for selecting multiple options in a list, i.e. "select all that apply"
- **Radio button** - used for selecting exactly one option in a list, i.e. "select one"
- **On/off switch** - used for enabling/disabling certain features/settings within an application, i.e. "dark mode, wifi, mobile data, etc"

"If it's hell for spinners, it's even hell-ier with radio buttons" - M. Fokaefs 2023

Dialogues

- Dialogues are used for information, alert, popups, feedback, etc. These should be used only for important prompts like info, settings, etc. since they put the current activity of the application on pause.
- **Toast**: A temporary lightweight popup that subtly alerts the user that something has happened. It automatically fades away.

Touch & multitouch

In android, every touch screen is referred to as a "gesture". However, we also want to discuss gestural inputs in a different contest (unistroke recognition).

Touch mode

- Android buttons are not focusable in touch mode, so we want to attach a `OnClickListener` instead of a `OnTouchListener`.