Layouts and Views

Repo: https://github.com/karlmorris/AndroidViewsAndLayouts
Overview

- Views and their uses
- Event listeners
- What are layouts
- Creating and using layouts
- Common layouts and examples
- Layout parameters
Views

• The basic building block of UI components
  – Widgets (buttons, images, etc.)
  – Layouts
• Subclass ViewGroup used for composite views
  – Layouts, Adapter Views, Toolbar, etc
• Responsible for **drawing** and **event handling**
Views

- Input Controls allow user input to your application
## Common Controls

<table>
<thead>
<tr>
<th>Control Type</th>
<th>Description</th>
<th>Related Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>A push-button that can be pressed, or clicked, by the user to perform an action.</td>
<td>Button</td>
</tr>
<tr>
<td>Text Field</td>
<td>An editable text field. You can use the AutoCompleteTextView widget to create a text entry widget that provides auto-complete suggestions.</td>
<td>EditText, AutoCompleteTextView</td>
</tr>
<tr>
<td>Checkbox</td>
<td>An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.</td>
<td>CheckBox</td>
</tr>
<tr>
<td>Radio Button</td>
<td>Similar to checkboxes, except that only one option can be selected in the group.</td>
<td>RadioGroup, RadioButton</td>
</tr>
<tr>
<td>Toggle Button</td>
<td>An on/off button with a light indicator.</td>
<td>ToggleButton</td>
</tr>
<tr>
<td>Spinner</td>
<td>A drop-down list that allows users to select one value from a set.</td>
<td>Spinner</td>
</tr>
<tr>
<td>Pickers</td>
<td>A dialog for users to select a single value for a set by using up/down buttons or via a swipe gesture. Use a DatePicker widget to enter the values for the date (month, day, year) or a TimePicker widget to enter the values for a time (hour, minute, AM/PM), which will be formatted automatically for the user's locale.</td>
<td>DatePicker, TimePicker</td>
</tr>
</tbody>
</table>
Layouts

• Define the visual structure for a user interface items such as
  – Activities
  – Application Widgets
  – Notifications
Creating a layout

• 2 methods
  – Describe UI elements in an XML layout file
    • During compile, XML resource is converted to a view
  – Instantiate layout elements at runtime
Advantages of declaring a layout

- Separation of code and presentation
- You can modify elements and their properties without modifying your code or writing more complex code
  - e.g. Defining for various screen sizes and orientation
- **Much** easier to visualize
XML Example

<?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"
    android:orientation="vertical" >
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
Load an XML resource

setContentView(R.layout.layout_file_name)
Common Layouts

Linear Layout

Relative Layout

Web View

<html>
  <!-- web page -->
</html>
Common Layouts

- Linear Layout
  - Organizes its children into a single horizontal or vertical row

- Relative Layout
  - Allows you to specify the location of the child object relative to each other or to the parent

- Web View
  - Displays web pages
Attributes

- ID – used to identify an object in a ViewGroup
  - Button myButton = (Button) findViewById(R.id.my_button);
- Layout position
- Size, Padding and Margins
- Layout Parameters...
Layout Parameters

• Define appropriate parameters for a view within its parent layout

• All layouts include an inner class that defines their respective parameters

• All layouts include a mandatory width and height setting. Others are generally optional.
Layout Parameters
Event Listeners

• An object with callback methods that are invoked when a specified event takes place

• Triggered by the Android framework when the user interacts with the widget
Event Listener Callbacks (View.On...Listener)

- **onClick**
  - Called when the user touches the item
- **onLongClick**
  - Called when the user touches and holds an item
- **onFocusChange**
  - Called when user navigates to or away from an item
- **onKey**
  - Called when the user presses or releases a hardware key
- **onTouch**
  - Called when the user performs any touch event
- **onCreateContextMenu**
  - Called when a context menu is being built