Mobile Application Development

Aileen Pierce

ANDROID USER INTERFACES

XML

- XML stands for eXtensible Markup Language
- XML is a markup language designed to structure data
- XML rules
  - First line - XML declaration
  - XML documents must have a root element
  - Attribute values must be in quotes
  - All elements must have a closing tag
  - Tags are case sensitive
  - Elements must be properly nested
  - XML must be well formed

View

- Every item in a user interface is a subclass of the Android View class **`android.view.View`**
  - Referred to as views, widgets, or components
- Views that can contain other views are subclassed from the Android ViewGroup class **`android.view.ViewGroup`** which is a subclass of the View class
  - Single parent view with multiple children

Linear Layout

- A linear layout displays views next to each other
- **`android:orientation`**
  - vertical: views are displayed in a single column
  - horizontal: views are displayed in a single row
- **`android:layout_width`** and **`android:layout_height`** are required attributes for all user interface elements
  - match_parent: view will be as big as its parent
  - wrap_content: view will be as big as its contents require

Linear Layout

- Linear layouts can be manipulated using different attributes on its components
- Allocating weight using **`layout_weight`** to a view tells it to stretch to take up extra space in the layout
  - First the layout ensures there is room for all the views
  - Then any extra space is divided proportionally between the views with a weight of 1 or greater
  - If only 1 view has a weight of 1 it will get all the extra space

Linear Layout

- The **`android:gravity`** attribute lets you specify how you want to position the contents of a view inside the view
  - top, bottom, left, right, and others
- The **`android:layout_gravity`** attribute lets you specify where you want a view in a linear layout to appear in its enclosing space.
  - top, bottom, left, right, and others
- Linear layouts use gravity instead of align so you can't use the android:layout_align attributes

User Interface

- All layouts and user interface components are subclasses of **`android.view.View`** so they all share common functionality
  - Getting and setting properties
    - **`findViewById()`**
  - Size and position
  - Event handling and listening
    - **`onClick`**

TextView

- Text views are used to display text
  `<TextView …/>`
- The **`android:textSize`** attribute controls the sixe of the text
  - Use the sp unit for scale-independent pixels
  - Scales based on the user's font size setting
- The **`setText(text)`** method changes the string in the text view

EditText

- Edit text is like a text view but editable
  `<EditText …/>`
- The **`android:hint`** attribute gives a hint to the user as to how to fill it in
- The **`android:inputType`** attribute defines what type of data you're expecting
  - Number, phone, textPassword, and others
  - Android will show the relevant keyboard
  - Can chain multiple input types with "|"
- **`getText().toString()`** retrieves the String

Button

- Buttons usually make your app do something when clicked **`<Button …/>`**
- Use the **`android:onClick`** attribute and assign it the name of the method you want to call in your activity code
- Your method must follow the notation

```
public void methodname(View view) { }
```

Toggle Button

- A toggle button allows the user to choose between two states **`<ToggleButton …/>`**
- The **`android:textOn`** attribute determines the text on the button when the state is ON
- The **`android:textOff`** attribute determines the text on the button when the state is OFF
- The **`isChecked()`** method returns a boolean – true if it's on, false if it's off

Switch

- A switch is a slider control that acts in the same way as a toggle button **`<Switch …/>`**
- The **`android:textOn`** and **`android:textOff`** attributes determine the text you want to display depending on the state of the switch
- The **`isChecked()`** method returns a boolean – true if it's on, false if it's off

CheckBox

- Check boxes let you display multiple options **`<CheckBox …/>`**
- Each check box is independent of the others
- The **`isChecked()`** method returns a boolean – true if it's checked, false if it's not

RadioButton

- Radio buttons let you display multiple options
  `<RadioButton …/>`
- Radio buttons must be part of a radio group that are dependent on the others
  `<RadioGroup …/>`
- Users can select only ONE radio button
- The **`getCheckedRadioButtonId()`** method returns the id(integer) of the chosen button
  - -1 means no button was chosen

Spinner

- A spinner presents a drop-down list of values from which only one can be selected
  `<Spinner …/>`
- You can store the values as an array in strings.xml
- The **`getSelectedItem()`** method returns the String of the selected item

ImageView

- Images are added to the res/drawable folder in your project
- The **`android:src`** attribute specifies what image you want to display
  - @drawable/imagename
- The **`android:contentDescription`** attribute holds a string description of the image
- The **`setImageResource()`** method sets the image source
- The **`setContentDescription()`** method sets the description

ScrollView

- A scrollview adds a vertical scrollbar to your layout **`<ScrollView>`**
  - Move the attributes from the original layout root to the scrollview as it is now the root
- Horizontal scrollbars are also available **`<HorizontalScrollView>`**

Launcher Icons

- Launcher(app) icons should be provided
  - mdpi: 48x48 px (baseline)
  - hdpi: 72x72 px
  - xhdpi: 96x96 px
  - xxhdpi: 144x144 px
  - Xxxhdpi: 192x192 px
- Launcher icons named ic_launcher.png go into density specific res/mipmap folders (i.e. res/mipmap-mdpi)
- Launcher icons should be designed specifically for Android. Avoid mimicking visual elements and styles from other platforms.