Android XML Layout and Java Explanation

Qiong LIU

October 14, 2024

1 Introduction

This document explains the structure and functionality of an Android XML layout file, specifically focusing on the use of `LinearLayout`, `TextView`, and `Button` elements. It also shows how to incorporate Java code with predefined formatting for better readability.

Further, we explain the Android system structure.

2 Android XML Code Breakdown

2.1 XML Namespace Declaration

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">
```

- `<?xml version="1.0" encoding="utf-8"?>` This line declears that the document is an XML file and specifies the XML version and character encoding.
- `<androidx.constraintlayout.widget.ConstraintLayout>:` The root layout for this UI, offering flexible constraint-based layouts.
- `xmlns:android="http://schemas.android.com/apk/res/android"` namespace for Android's built-in attributes.
- `mlns:app="http://schemas.android.com/apk/res-auto":` This namespace is used for custom attributes related to app-specific features
- `xmlns:tools="http://schemas.android.com/tools"` This namespace is used for attributes that are only relevant for layout previews in Android Studio.

2.2 LinearLayout Properties

```
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
```

These lines define layout properties for the `LinearLayout`:

- `android:orientation="vertical"`: Specifies that child views inside the `LinearLayout` should be stacked vertically.(android:orientation="horizontal", show from left to right.)
- `android:layout_width="match_parent"`: The `LinearLayout` will expand to match the full width of its parent view.
- `android:layout_height="match_parent"`: The `LinearLayout` will expand to match the full height of its parent view.

```
      app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
```

- `app:layout_constraintBottom_toBottomOf="parent` Aligns the bottom of the view with the bottom of its parent (ConstraintLayout).
- `app:layout_constraintEnd_toEndOf="parent"` aligns the right of the view with the right side of its parent
- `app:layout_constraintStart_toStartOf="parent"` Algins the left
- `app:layout_constraintTop_toTopOf="parent"` Aligns the top with the top of its parent

2.3 TextView Definition

```
<TextView
    android:id="@+id/myTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!" />
```

This block defines a `TextView` element, which displays text on the screen:

- `android:id="@+id/myTextView"`: Assigns a unique ID to the `TextView`, allowing it to be referenced in the code.
- `android:layout_width="wrap_content"`: The width of the `TextView` will be just enough to fit the content.

- `android:layout_height="wrap_content"`: The height of the `TextView` will adjust to fit the content.
- `android:text="Hello World!"`: Sets the text displayed in the `TextView` to ”Hello World!”.

2.4 Button Definition

```
<Button
    android:id="@+id/myButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Click Me" />
```

This block defines a `Button` element:

- `android:id="@+id/myButton"`: Assigns a unique ID to the `Button`, allowing it to be referenced in the code.
- `android:layout_width="wrap_content"`: The width of the `Button` will adjust to fit its content.
- `android:layout_height="wrap_content"`: The height of the `Button` will also adjust to fit its content.
- `android:text="Click Me"`: Sets the text displayed on the `Button` to ”Click Me”.

3 Java Code Example

Below is an example of a simple Java class for an Android activity that inflates the layout defined above.

```
package com.example.myapplication;
// Package declaration for the application.

import android.os.Bundle;
// Import for managing activity state using the Bundle class.

import androidx.activity.EdgeToEdge;
// Import the EdgeToEdge helper class, used to enable edge-to-edge
    display, i.e., full screen.

import androidx.appcompat.app.AppCompatActivity;
// Import for AppCompatActivity, which provides backward
    compatibility for modern Android features. (The ability of
    software to run on older versions of the operating system or
    hardware while still using newer features or APIs)

import androidx.core.graphics.Insets;
// Import for Insets class, which holds inset values for window
    system bars like status and navigation bars.
```

```java
import androidx.core.view.ViewCompat;
// Import for ViewCompat, used to work with views in a backward-
    compatible way.

import androidx.core.view.WindowInsetsCompat;
// Import for WindowInsetsCompat, which provides information about
    system windows and their insets.

public class MainActivity extends AppCompatActivity {
    // Main class extending AppCompatActivity for compatibility
        across different Android versions.

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // onCreate method, called when the activity is first
            created.
        //A Bundle is a data structure that stores key-value pairs.
        //When the activity is first created, savedInstanceState is
             null; If the activity is being recreated after being
            previously destroyed, savedInstanceState contains the
            data saved in the onSaveInstanceState() method.

        super.onCreate(savedInstanceState);
        // Call the superclass's onCreate method to initialize the
            activity.

        EdgeToEdge.enable(this);
        // Enable edge-to-edge mode, allowing the app to use the
            full screen (including under the system bars).

        setContentView(R.layout.activity_main);
        // Set the content view to the layout file 'activity_main.
            xml'.

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id
            .main), (v, insets) -> {
            // Set an OnApplyWindowInsetsListener to adjust the
                view when system window insets (e.g., status bar,
                navigation bar) are applied.

            Insets systemBars = insets.getInsets(WindowInsetsCompat
                .Type.systemBars());
            // Get the insets for system bars (status bar,
                navigation bar) using the WindowInsetsCompat API.

            v.setPadding(systemBars.left, systemBars.top,
                systemBars.right, systemBars.bottom);
            // Set padding for the view based on the system bar
                insets, ensuring the content is adjusted to avoid
                being overlapped.

            return insets;
        });
    }
}
```

4 Android System Architecture

- Applications: these are the apps you interact with directly on your phone, such as Contacts, Phone, and the Browser. They provide the services and features you use every day.
- Application Framework: this layer provider Android developers with the building blocks need to create app. It includes components to manage activities, windows, notifications and access to content.
- Libraries: Essential tools that handle specific tasks. For example:
  1. OpenGL: for rendering 3D graphics;
  2. SQLite: for database management;
  3. WebKit: for web browsing;
- Android Runtime (Dalvik/ART): The environment where your apps run. It converts the app code into machine language that the device understand (complication).
- Linux Kernel: the foundation of the Android system. It manage the hardware (like memory and CPU) and provides low-level system services. Also the drivers and power management.