

Rajdip Sanyal (12/09/2024)
sanyalrajdip864@gmail.com

Android Framework With Java

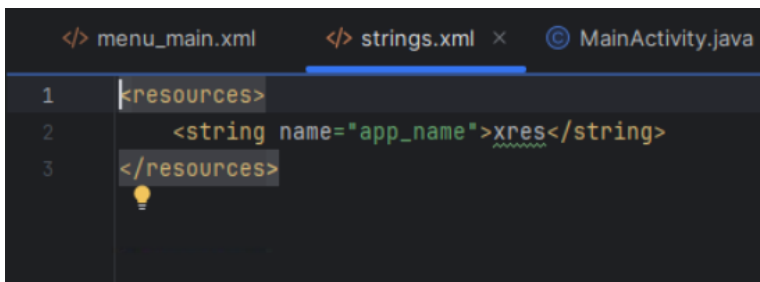
Using The Manifest Editor in Android

The AndroidManifest.xml file is a crucial component of any android application. It serves as the central configuration file for our app, providing essential information about the app to the Android system. Using the manifest editor in android studio can simplify the process of editing and managing this file.

Externalizing and Creating Resources in Android

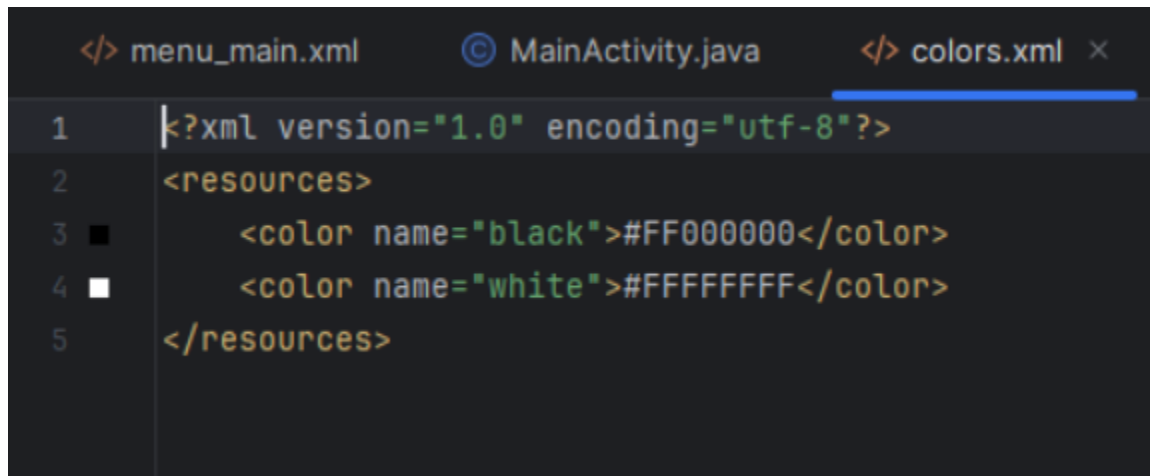
In android development, externalizing and creating resources refers to managing and organizing various assets and configurations for an app in a way that makes it more modular, maintainable, and adaptable. Here's a breakdown of what this involves:

1. **Externalizing Resources**: Externalizing resources means separating them from the code and placing them in dedicated resource files. This helps keep the code clean and allows for easier updates and localization.
Strings: Store all text in res/values/strings.xml rather than hard-coding strings in the layout or code files. This makes it easier to manage and translate text.



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">xres</string>
</resources>
```

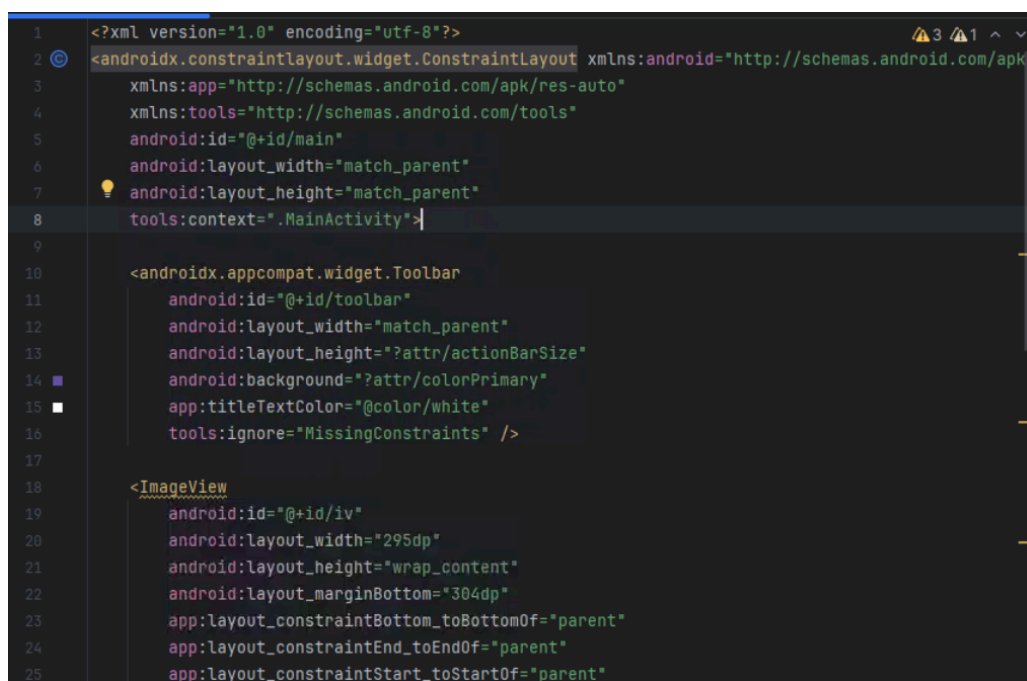
Colors: Store all text in res/values/colors.xml rather than hard-coding colors in the layout or code files. This makes it easier to manage and translate text.



```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

2. Creating Resources: Creating resources involves defining and organizing these elements so they can be used effectively in our application.

Create Layouts: Use XML files in res/layout to design the UI for different screen sizes and orientations. Each layout file represents a different screen or component of our app.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:titleTextColor="@color/white"
        tools:ignore="MissingConstraints" />

    <ImageView
        android:id="@+id/iv"
        android:layout_width="295dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="304dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent">
```

Styles and Themes in Android

In android development, styles and themes are crucial for defining the visual appearance of an app. They help us to maintain consistency across different screens and provide a way to easily update the look and feel of our application.

Styles: A style in Android is a collection of properties that define the appearance of a user interface (UI) element. For example, we can set the font size, color, padding, margins, and more for UI elements like TextViews, Buttons, etc.

```
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
18 <androidx.appcompat.widget.Toolbar
15     app:titleTextColor="@color/white"
16     tools:ignore="MissingConstraints" />
17
18 <ImageView
19     android:id="@+id/iv"
20     android:layout_width="295dp"
21     android:layout_height="wrap_content"
22     android:layout_marginBottom="304dp"
23     app:layout_constraintBottom_toBottomOf="parent"
24     app:layout_constraintEnd_toEndOf="parent"
25     app:layout_constraintStart_toStartOf="parent"
26     app:srcCompat="@drawable/xresimg2" />
27
28 <Button
29     android:id="@+id/anim"
30     android:layout_width="wrap_content"
31     android:layout_height="wrap_content"
32     android:text="Blink Animate"
33     app:layout_constraintBottom_toBottomOf="parent"
34     app:layout_constraintEnd_toEndOf="parent"
35     app:layout_constraintStart_toStartOf="parent"
36     app:layout_constraintTop_toBottomOf="@+id/iv" />
37
```

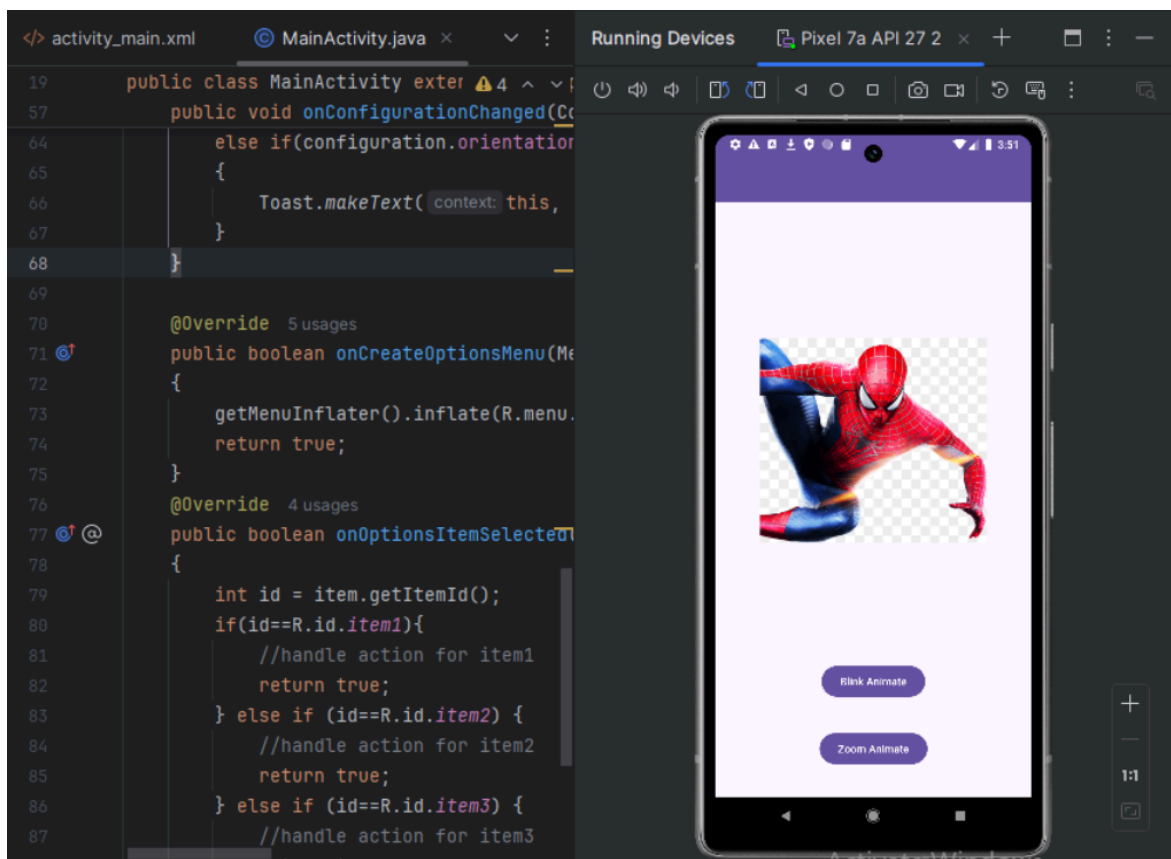
Themes: A theme is a collection of styles that we apply to an entire activity or application. It dictates the overall appearance of the app, including things like colors, fonts, and general styling of UI components.

```
1 <resources xmlns:tools="http://schemas.android.com/tools">
2     <!-- Base application theme. -->
3     <style name="Base.Theme.Xres" parent="Theme.Material3.DayNight.NoActionBar">
4         <!-- Customize your light theme here. -->
5         <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
6     </style>
7
8     <style name="Theme.Xres" parent="Base.Theme.Xres" />
9 </resources>
```

Using Animations in Android

Animations in Android development are used to enhance the user experience by making interactions smoother and more engaging. They help make apps feel more responsive and dynamic. Here's an overview of the different types of animations and how they are typically used in Android development:

1. Alpha Animation: Controls the transparency of a view (e.g.fading in or out).
2. Scale Animation: Changes the size of a view (e.g.zooming in or out).
3. Translate Animation: Moves a view from one position to another (e.g. sliding left or right).
4. Rotate Animation: Rotates a view around its center or a specified point.



Creating Menus in Android

Creating menus in Android development involves designing and implementing various types of menus that users can interact with.

How to implement?

Define menu items in an XML file located in the res/menu directory (e.g., res/menu/main_menu.xml).

Override onCreateOptionsMenu(Menu menu) in your Activity to inflate the menu XML file.

Handle menu item clicks by overriding onOptionsItemSelected(MenuItem item).

Using Resources in Android

In android development resources are essential components that help create a rich and dynamic user experience. These resources include various types of files and configurations that can be used across our application. Here's a breakdown of the different types of resources and how they are used:

1. Drawable Resources: Drawable resources are graphics or images used in your app. They can be in various formats, including:

Bitmap files (e.g., PNG, JPG)

Vector drawables (e.g., SVG, XML-based)

Shape drawables (XML files defining shapes)

State list drawables (XML files defining different states)

We place these files in the res/drawable directory. Vector drawables are especially useful because they scale well on different screen sizes and densities.

2. Layout Resources: Layout resources define the structure of our user interface. They are XML files that describe the arrangement of UI elements on the screen. For example, we might use a LinearLayout, RelativeLayout, or ConstraintLayout to position your UI components. These files are stored in the res/layout directory. We typically create a layout XML file for each screen or fragment in our app.

3. String Resources: String resources are used to manage text in our app. They help with localization by allowing us to define text in XML files, which can then be translated into different languages. This ensures that our app can be easily adapted for different locales. String resources are placed in res/values/strings.xml.

4. Color Resources: Color resources define colors used throughout our app. This approach helps maintain consistency and makes it easier to update colors across our app. We define colors in XML files and refer to them in your layouts and drawables. Color resources are placed in res/values/colors.xml.

5. Menu Resources: Menu resources define the menu options for our app, such as options menus, context menus, and popup menus. These are XML files that specify the menu items and their associated actions. Menu resources are stored in the res/menu directory.

6. Animation Resources: Animation resources define animations and transitions used in our app. We can create frame-by-frame animations or property animations that change the properties of UI elements over time. Animation resources are defined in XML files and placed in the res/anim directory.

7. Raw Resources: Raw resources are files that are stored in their raw form and accessed directly as a stream of data. This includes files such as audio files, video files, or custom data files that we want to include in our app. Raw resources are placed in the res/raw directory.

8. XML Resources: In addition to the aforementioned types, XML resources can include configuration files, such as res/xml which is used for defining XML-based resources that are not layouts or drawables, like preferences or settings.