Android Development

04 Practical - Text and scrolling View

# App Overview

The Scrolling Text app demonstrates the ScrollView UI component. ScrollView is a ViewGroup that in this example contains a TextView. It shows a lengthy page of text—in this case, a music album review—that the user can scroll vertically to read by swiping up and down. A scroll bar appears in the right margin. The app shows how you can use text formatted with minimal HTML tags for setting text to bold or italic, and with new-line characters to separate paragraphs. You can also include active web links in the text.

A screenshot of a social media post

Description automatically generated

In the above figure, the following appear:

1. An active web link embedded in free-form text
2. The scroll bar that appears when scrolling the text

# Add and edit TextView elements

You will create an Android project for the ScrollingText app, add TextView elements to the layout for an article title and subtitle, and change the existing "Hello World" TextView element to show a lengthy article. The figure below is a diagram of the layout.

A screenshot of a cell phone

Description automatically generated

You will make all these changes in the XML code and in the strings.xml file. You will edit the XML code for the layout in the Text pane, which you show by clicking the **Text** tab, rather than clicking the **Design** tab for the Design pane. Some changes to UI elements and attributes are easier to make directly in the Text pane using XML source code.

## Create the project and the TextView elements

Create the project and the TextView elements and use TextView attributes for styling the text and background.

1. In Android Studio create a new project with the following parameters:

|  |  |
| --- | --- |
| **Attribute** | **Value** |
| Application Name | Scrolling Text |
| Company Name | android.example.com (or your own domain) |
| Phone and Tablet Minimum SDK | Use default |
| Template | Empty Views Activity |
| Generate Layout File checkbox | Selected |
| Backwards Compatibility (AppCompat) checkbox | Selected |

1. In the **app > res > layout** folder in the **Project > Android** pane, open the **activity\_main.xml** file, and click the **Text** tab to see the XML code.

At the top, or root, of the View hierarchy is the ConstraintLayout ViewGroup:

android.support.constraint.ConstraintLayout

1. Change this ViewGroup to [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html). The second line of code now looks something like this:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

RelativeLayout lets you place UI elements relative to each other, or relative to the parent RelativeLayout itself.

The default "Hello World" TextView element created by the Empty Layout template still has constraint attributes (such as app:layout\_constraintBottom\_toBottomOf="parent"). Don't worry—you will remove them in a subsequent step.

1. Delete the following line of XML code, which is related to ConstraintLayout:

xmlns:app="http://schemas.android.com/apk/res-auto"

The block of XML code at the top now looks like this:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.android.scrollingtext.MainActivity">

1. Add a TextView element above the "Hello World" TextView by entering **<TextView**. A TextView block appears that ends with /> and shows the layout\_width and layout\_height attributes, which are required for the TextView.
2. Enter the following attributes for the TextView. As you enter each attribute and value, suggestions appear to complete the attribute name or value.

|  |  |
| --- | --- |
| **TextView #1 attribute** | **Value** |
| android:layout\_width | "match\_parent" |
| android:layout\_height | "wrap\_content" |
| android:id | "@+id/article\_heading" |
| android:background | "@color/colorPrimary" |
| android:textColor | "@android:color/white" |
| android:padding | "10dp" |
| android:textAppearance | "@android:style/TextAppearance.DeviceDefault.Large" |
| android:textStyle | "bold" |
| android:text | "Article Title" |

1. Extract the string resource for the android:text attribute's hardcoded string "Article Title" in the TextView to create an entry for it in **strings.xml**.  
     
   Place the cursor on the hardcoded string, press Alt-Enter (Option-Enter on the Mac) and select **Extract string resource**. Make sure that the **Create the resource in directories** option is selected, and then edit the resource name for the string value to **article\_title**.
2. Extract the dimension resource for the android:padding attribute's hardcoded string "10dp" in the TextView to create dimens.xml and add an entry to it.  
     
   Place the cursor on the hardcoded string, press Alt-Enter (Option-Enter on the Mac), and select **Extract dimension resource**. Make sure that the **Create the resource in directories** option is selected, and then edit the Resource name to **padding\_regular**.
3. Add another TextView element above the "Hello World" TextView and below the TextView you created in the previous steps. Add the following attributes to the TextView:

|  |  |
| --- | --- |
| **TextView #2 Attribute** | **Value** |
| layout\_width | "match\_parent" |
| layout\_height | "wrap\_content" |
| android:id | "@+id/article\_subheading" |
| android:layout\_below | "@id/article\_heading" |
| android:padding | "@dimen/padding\_regular" |
| android:textAppearance | "@android:style/TextAppearance.DeviceDefault" |
| android:text | "Article Subtitle" |

Because you extracted the dimension resource for the "10dp" string to padding\_regular in the previously created TextView, you can use "@dimen/padding\_regular" for the android:padding attribute in this TextView.

1. Extract the string resource for the android:text attribute's hardcoded string "Article Subtitle" in the TextView to article\_subtitle.
2. In the "Hello World" TextView element, delete the layout\_constraint attributes:

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

1. Add the following TextView attributes to the "Hello World" TextView element, and change the android:text attribute:

|  |  |
| --- | --- |
| **TextView Attribute** | **Value** |
| android:id | "@+id/article" |
| android:layout\_below | "@id/article\_subheading" |
| android:lineSpacingExtra | "5sp" |
| android:padding | "@dimen/padding\_regular" |
| android:text | Change to "Article text" |

1. Extract the string resource for "Article text" to article\_text and extract the dimension resource for "5sp"to line\_spacing.
2. Reformat and align the code by choosing Code > Reformat Code. It is a good practice to reformat and align your code so that it is easier for you and others to understand.

## Add the text of the article

In a real app that accesses magazine or newspaper articles, the articles that appear would probably come from an online source through a content provider or might be saved in advance in a database on the device.

For this practical, you will create the article as a single long string in the strings.xml resource.

1. In the **app > res > values** folder, open **strings.xml**.
2. Open any text file with a large amount of text.
3. Enter the values for the strings article\_title and article\_subtitle with either a made-up title and subtitle or use the values in the strings.xml file of the finished ScrollingText app. Make the string values single-line text without HTML tags or multiple lines.
4. Enter or copy and paste text for the article\_text string.

You can use the text in your text file, or use the text provided for the article\_text string in the strings.xml file of the finished ScrollingText app. The only requirement for this task is that the text must be long enough so that it doesn't fit on the screen.

# Run the App

Run the app. The article appears, but the user can't scroll the article because you haven't yet included a ScrollView (which you will do next).

Note also that tapping a web link does not currently do anything. You will also fix this next.

A screenshot of a cell phone

Description automatically generated

# Fix Stuff - Add a ScrollView and an active web link

## Add the autoLink attribute for active web links

Add the android:autoLink="web" attribute to the article TextView. The XML code for this TextView now looks like this:

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/article"

android:autoLink="web"

android:layout\_below="@id/article\_subheading"

android:lineSpacingExtra="@dimen/line\_spacing"

android:padding="@dimen/padding\_regular"

android:text="@string/article\_text" />

## Add a ScrollView to the layout

To make a View (such as a TextView) scrollable, embed the View inside a ScrollView.

1. Add a ScrollView between the article\_subheading TextView and the article TextView. As you enter **<ScrollView**, Android Studio automatically adds </ScrollView> at the end, and presents the android:layout\_width and android:layout\_height attributes with suggestions.
2. Choose **wrap\_content** from the suggestions for both attributes.

The code for the two TextView elements and the ScrollView now looks like this:

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/article\_subheading"

android:layout\_below="@id/article\_heading"

android:padding="@dimen/padding\_regular"

android:text="@string/article\_subtitle"

android:textAppearance=

"@android:style/TextAppearance.DeviceDefault"/>

<ScrollView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"></ScrollView>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/article"

android:autoLink="web"

android:layout\_below="@id/article\_subheading"

android:lineSpacingExtra="@dimen/line\_spacing"

android:padding="@dimen/padding\_regular"

android:text="@string/article\_text" />

1. Move the ending </ScrollView> code after the article TextView so that the article TextView attributes are entirely inside the ScrollView.
2. Remove the following attribute from the article TextView and add it to the ScrollView:

android:layout\_below="@id/article\_subheading"

With the above attribute, the ScrollView element will appear below the article subheading. The article is inside the ScrollView element.

1. Choose **Code > Reformat Code** to reformat the XML code so that the article TextView now appears indented inside the <ScrollView code.
2. Click the **Preview** tab on the right side of the layout editor to see a preview of the layout.

The layout now looks like the right side of the following figure:

A screenshot of a cell phone

Description automatically generated

# Run the App

To examine how the text scrolls:

1. Run the app on a device or emulator.

Swipe up and down to scroll the article. The scroll bar appears in the right margin as you scroll.

Tap the web link to go to the web page. The android:autoLink attribute turns any recognizable URL in the TextView (such as www.rockument.com) into a web link.

1. Rotate your device or emulator while running the app. Notice how the scrolling view widens to use the full display and still scrolls properly.
2. Run the app on a tablet or tablet emulator. Notice how the scrolling view widens to use the full display and still scrolls properly.