Computer Science Large Practical: Storage, Settings and Layouts

Stephen Gilmore
School of Informatics

October 26, 2017

Contents

Storing information

Storage on Android devices

- The Android framework provides several ways to store information on an Android device and have it persist between different runs of an app.
- One use might be to store a file locally, and a familiar file system is available for this which is accessible via the java.io.File API.
- If more structured data needs to be stored and queried then a SQLite database can be created and operated using the android.database.sqlite package.
- For storing small items of information across sessions with an app the android.content.SharedPreferences framework is the most appropriate.

Making values persistent — SharedPreferences

- SharedPreferences enables you to store key-value pairs; it can be used to save values of type Boolean, Float, Int, Long, String, and String Set.
- Preference files can be named, if you need more than one.
- Stored values from a previous session can be restored in the current session in the onCreate method.
- Updated values from the current session can be written in the onStop method.
- Any value can be considered a preference: it doesn't have to be user preferences (such as *"Sounds: on/off"* etc).

Example: A simple counter activity for button clicks

```
import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.content_main.*
class MainActivity : AppCompatActivity() {
    private var clicks = 0 // int counter for button clicks
    val PREFS_FILE = "MyPrefsFile" // for storing preferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setSupportActionBar(toolbar)
        button.setOnClickListener { _ -> clicks++ } // count one click
        fab.setOnClickListener { v ->
            Snackbar.make(v, "Clicks: $clicks", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show()
        }
    }
    ...
}
```

Reading in saved preferences [MainActivity ▸ onStart]

```
override fun onStart() {
    super.onStart()

    // Restore preferences
    val settings = getSharedPreferences(PREFS_FILE,
        Context.MODE_PRIVATE)

    // use 0 as the default value (this might be the first time the app is run)
    clicks = settings.getInt("storedClicks", 0)
}
```

Writing out updated preferences [MainActivity ▸ onStop]

```
override fun onStop() {
    super.onStop()

    // All objects are from android.context.Context
    val settings = getSharedPreferences(PREFS_FILE,
        Context.MODE_PRIVATE)

    // We need an Editor object to make preference changes.
    val editor = settings.edit()
    editor.putInt("storedClicks", clicks)

    // Apply the edits!
    editor.apply()
}
```

onStop is called even if our app is killed by the operating system

Lifetime of stored data

- Our data will persist in a storage area on our device which is private to our app.
- If we update our app and install a new APK on our device the data will persist between these app updates.
- Our data will persist until the app is uninstalled from the device.

Kotlin compilation

Aside: a note on Kotlin compilation

If your Kotlin project is taking a long time to compile in Android Studio (e.g. 10 minutes), and particularly if you receive the following message:

> *Could not perform incremental compilation: Could not connect to Kotlin compile daemon*

Then it could be worthwhile to update your gradle.properties file with the following settings:

```
# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. More details, visit
# http://www.gradle.org/docs/current/userguide/multi_project_builds.html
org.gradle.daemon=false
org.gradle.parallel=false
kotlin.incremental=false
```

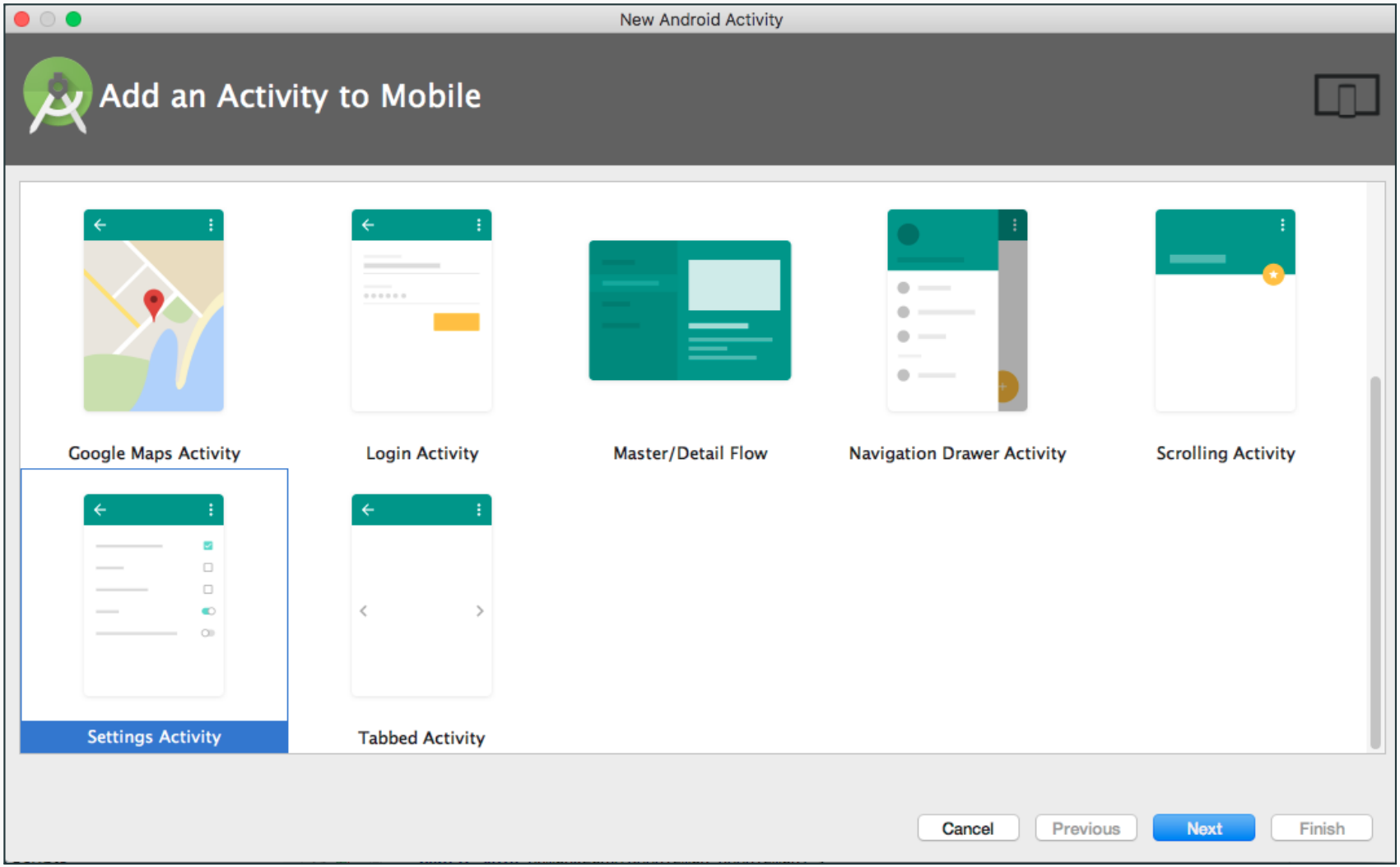
Settings

Adding a user interface component for settings

- Apps which allow users to set preferences need a user interface component to allow the users to edit these preferences.
- These are built on the android.preference.Preference API.
- The Preference class has subclasses including CheckBoxPreference, EditTextPreference, ListPreference, SwitchPreference, and others.

- Settings of these subclasses are declared in XML files stored in the XML resources folder (res/xml).
- These XML files reference strings (in res/values/strings.xml) and icons (in res/drawable).

Creating a new SettingsActivity



IDE code generation

- As we have seen already, adding a new Activity to our project in Android Studio will add a new source code file to our project but it may also do several other things, such as create new XML files, add image files, and update the Gradle build file.
- Most modern IDEs will generate code to automate common tasks which are likely to occur in multiple projects.
- This can be helpful, especially when we are unfamiliar with libraries and APIs which are used but code generation by IDEs has both advantages and disadvantages . . .

Pros and cons of generated code

Pro: We get a number of files in our project updated automatically with their interdependencies.

Con: If we later decide that we don't want this feature then it falls to us to manually remove these interdependent files.

Pro: We get a working example of an Activity added to our project.

Con: We might get a lot of code added to our project which we then have to understand and modify.

Pro: The generated code can encourage consistency between apps from different developers.

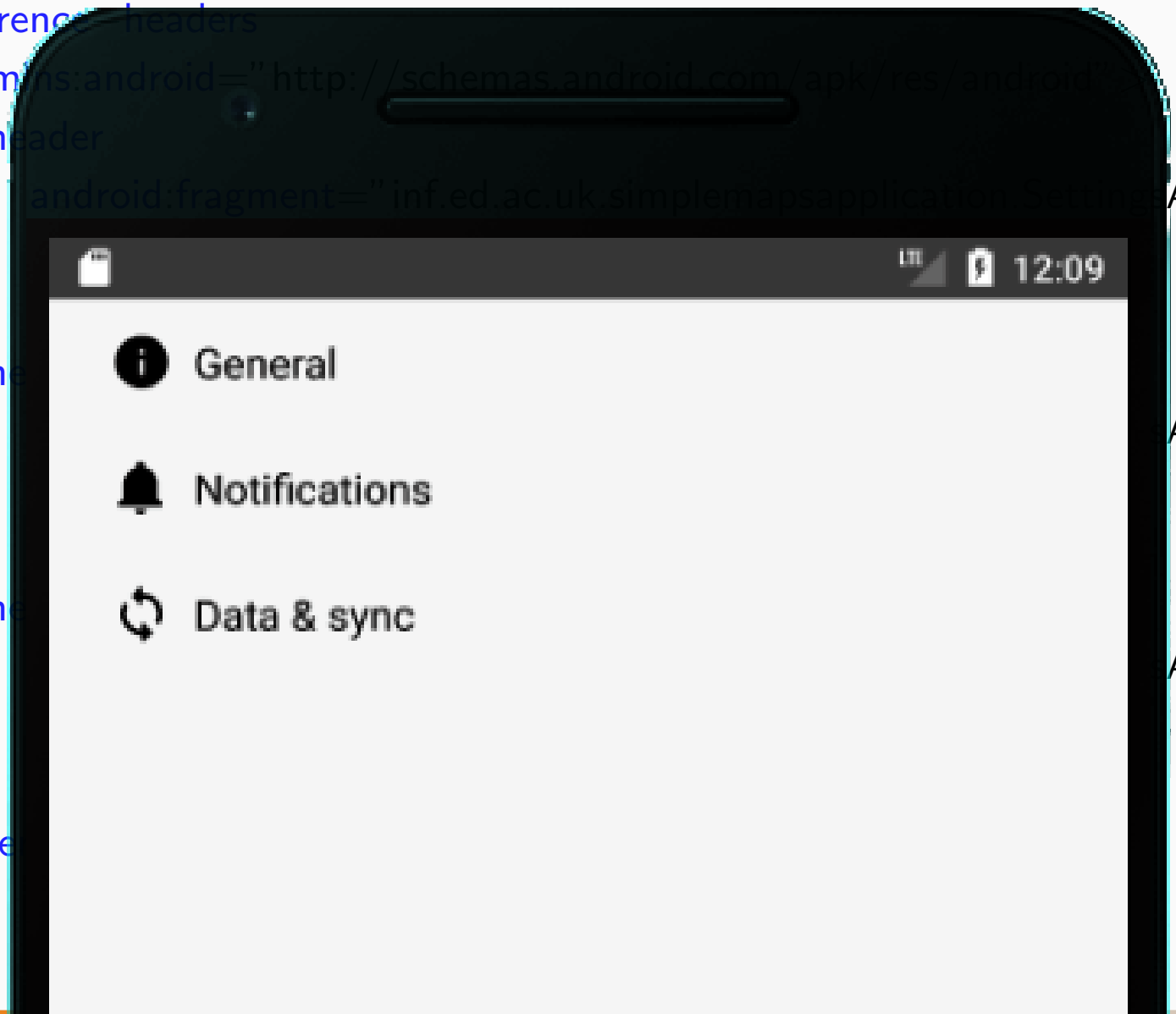
Con: This feels like it is really someone else's priority.

Generated code: Top-level default preferences

```
<!-- res/xml/preference_headers.xml -->
<preference-headers
    xmlns:android="http://schemas.android.com/apk/res/android">
    <header
        android:fragment="inf.ed.ac.uk.simplemapsapplication.SettingsActivity$Genera
        android:icon="@drawable/ic_info_black_24dp"
        android:title="@string/pref_header_general" />
    <header
        android:fragment="inf.ed.ac.uk.simplemapsapplication.SettingsActivity$Notific
        android:icon="@drawable/ic_notifications_black_24dp"
        android:title="@string/pref_header_notifications" />
    <header
        android:fragment="inf.ed.ac.uk.simplemapsapplication.SettingsActivity$DataS
        android:icon="@drawable/ic_sync_black_24dp"
        android:title="@string/pref_header_data_sync" />
</preference-headers>
```

Generated code: Top-level default preferences

```
<!-- res/xml/preference_headers.xml -->
<preference-headers
    xmlns:android="http://schemas.android.com/apk/res/android">
    <header
        android:fragment="inf.ed.ac.uk.simplemapsapplication.SettingsActivity$General
    <header
        android:fragment="inf.ed.ac.uk.simplemapsapplication.SettingsActivity$Notific
    <header
        android:fragment="inf.ed.ac.uk.simplemapsapplication.SettingsActivity$DataS
</prefe
```



Default general preferences [in res/xml/pref_general.xml]

```
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">

    <SwitchPreference
        android:defaultValue="true"
        android:key="example_switch"
        android:summary="@string/pref_description_social_recommendations"
        android:title="@string/pref_title_social_recommendations" />

    <EditTextPreference
        android:capitalize="words"
        android:defaultValue="@string/pref_default_display_name"
        ...
        android:title="@string/pref_title_display_name" />

    <ListPreference
        android:defaultValue="-1"
        android:entries="@array/pref_example_list_titles"
```

Default general preferences [in res/xml/pref_general.xml]

<PreferenceScreen

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

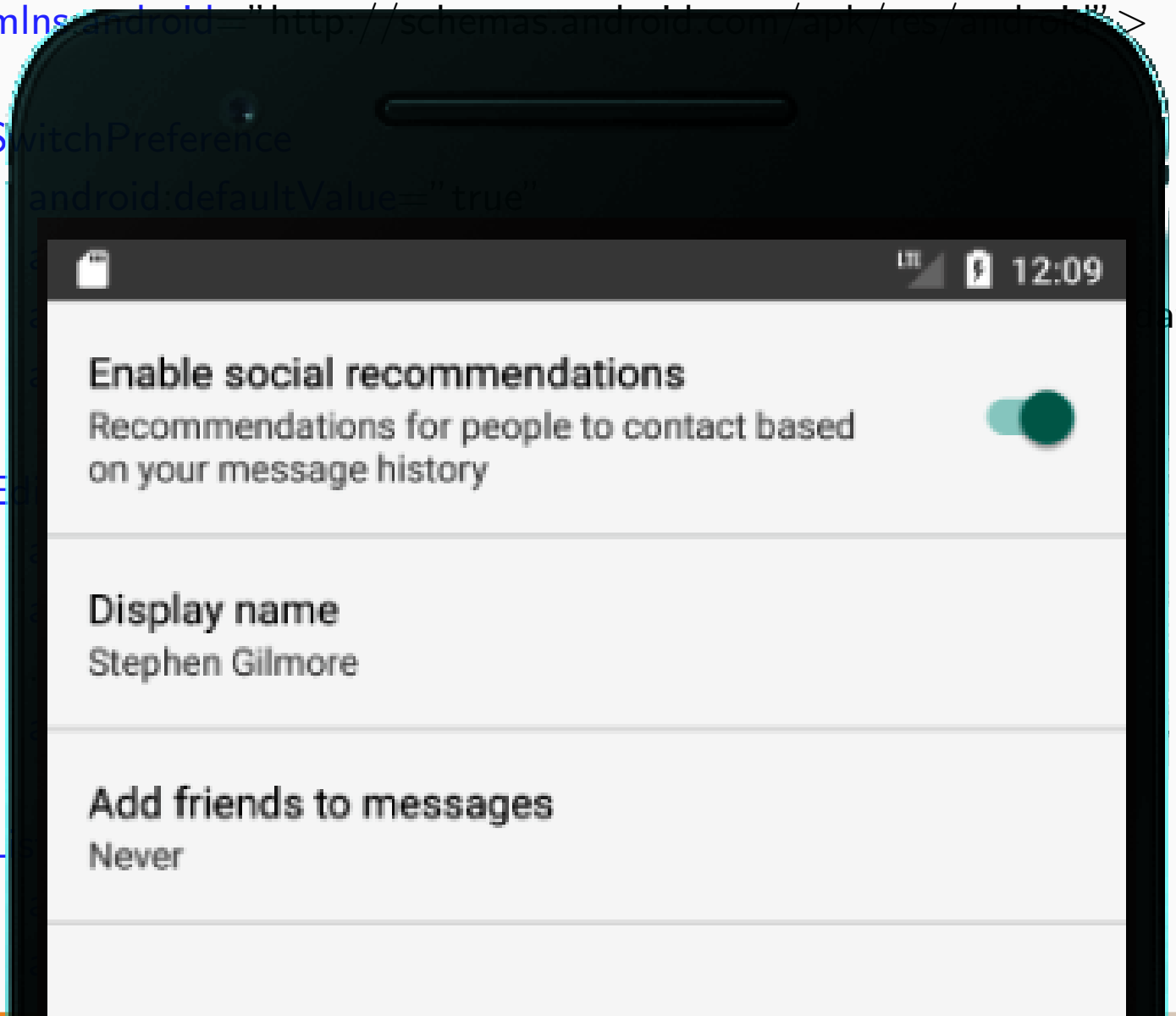
<SwitchPreference

android:defaultValue="true"

ations"

<Ed

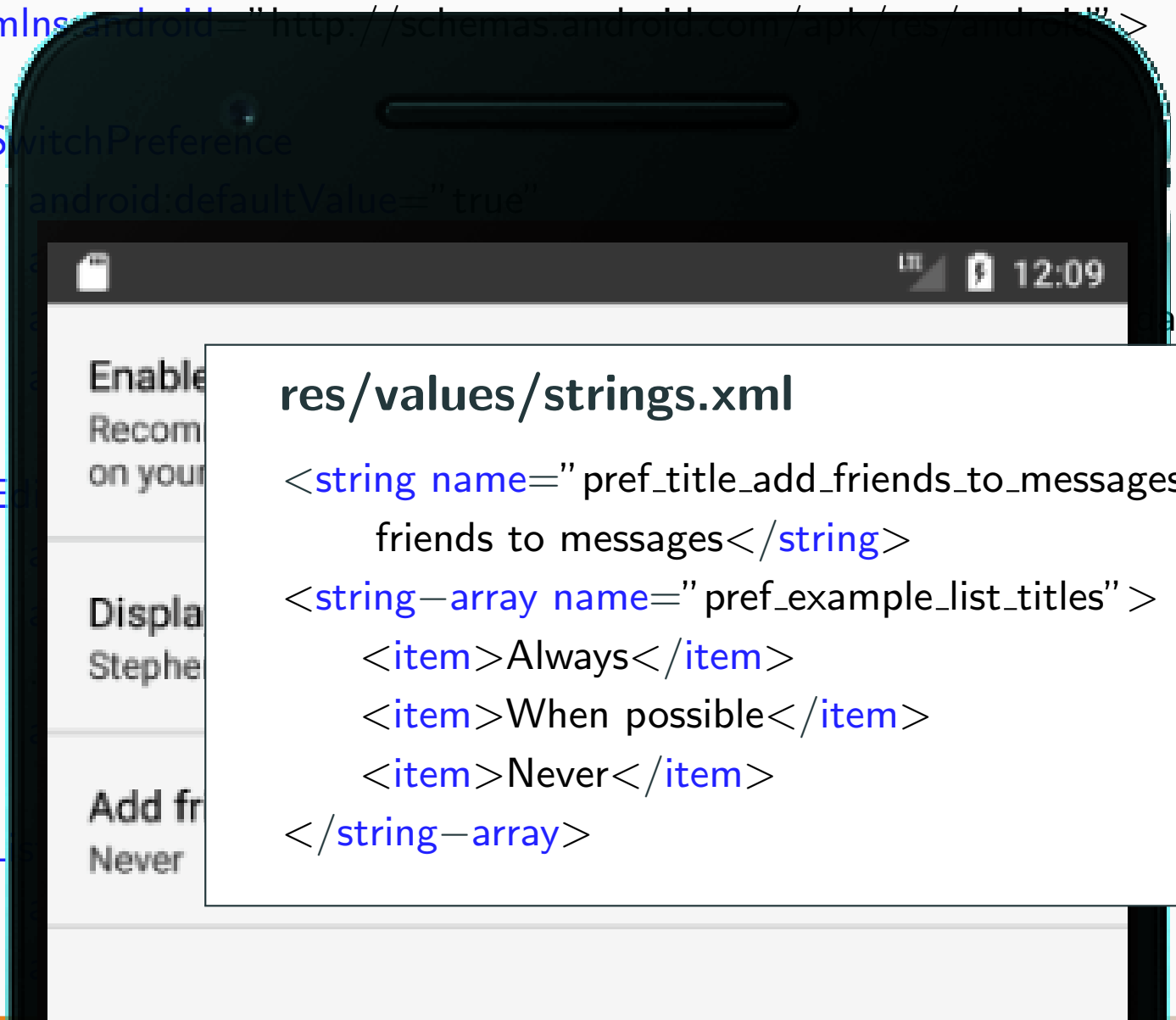
<L



Default general preferences [in res/xml/pref_general.xml]

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

<SwitchPreference
android:defaultValue="true"



res/values/strings.xml

```
<string name="pref_title_add_friends_to_messages">Add
    friends to messages</string>
<string-array name="pref_example_list_titles">
    <item>Always</item>
    <item>When possible</item>
    <item>Never</item>
</string-array>
```

Updating settings

- Preferences are saved in a SharedPreferences file.
- Every time that the user updates a preference via the user interface, the corresponding preference in SharedPreferences is updated automatically.

Attaching the SettingsActivity to the menu

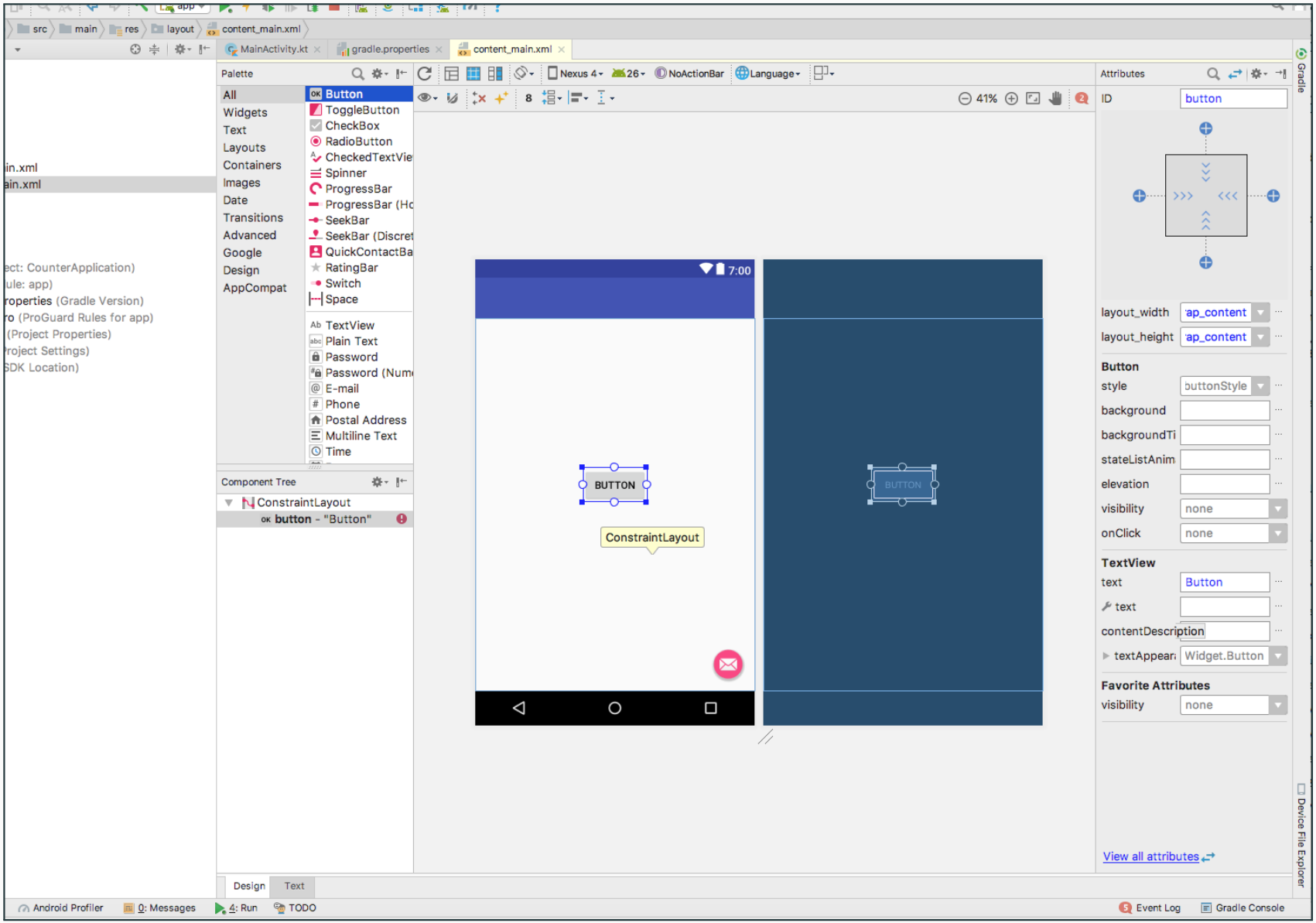
```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    return when (item.itemId) {
        R.id.action_settings -> {
            // User chose the "Settings" item, show the app settings UI
            val intent = Intent(this, SettingsActivity::class.java)
            startActivity(intent)
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}
```

Layouts

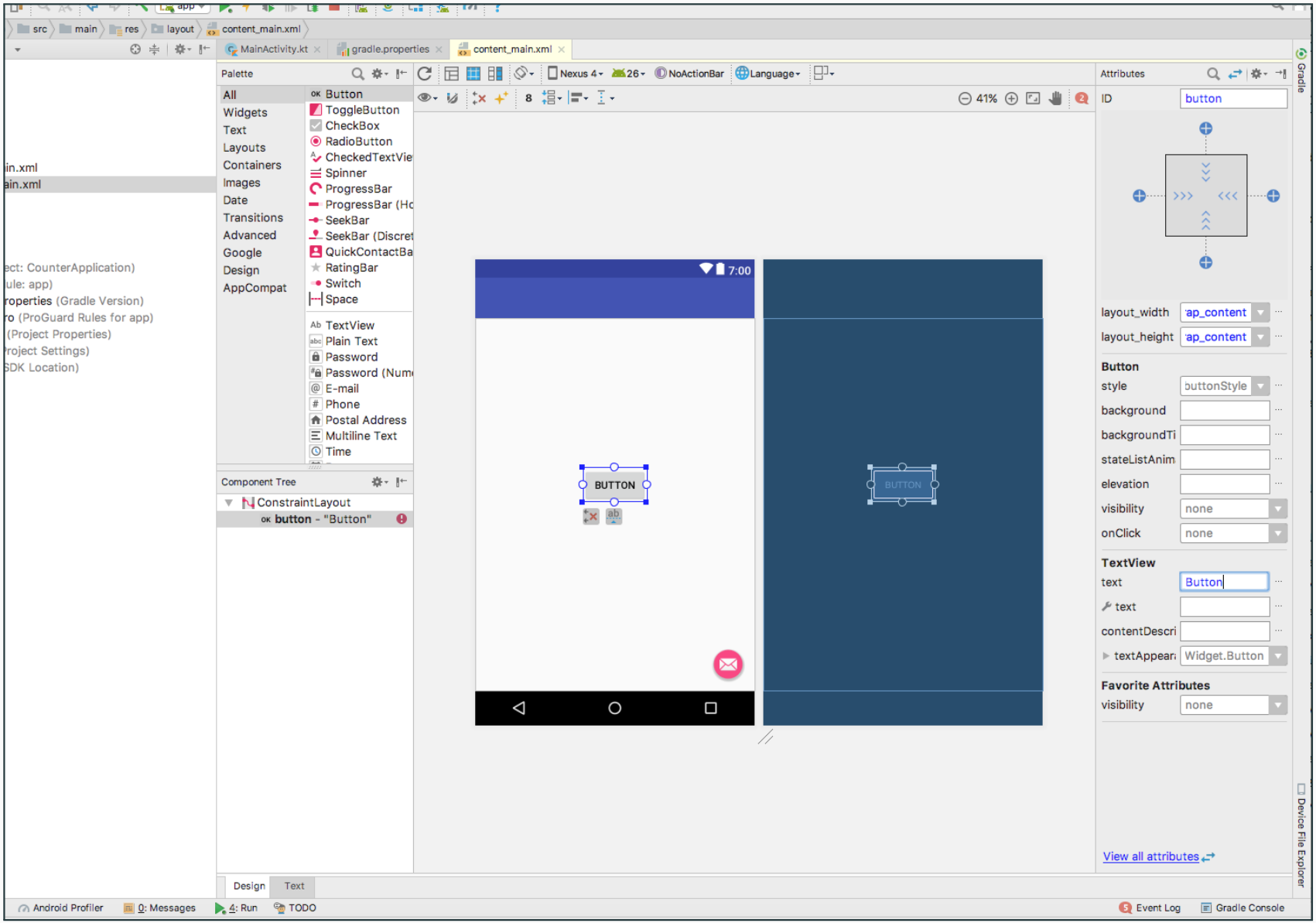
Producing layouts with ConstraintLayout

- When you want to design a layout of your own for your activity, and not base it on an existing design, the best tool for this purpose is the Layout Editor in Android Studio.
- When used together with the ConstraintLayout style, this allows you to build a user interface entirely by drag-and-dropping widgets onto the canvas instead of editing the XML.
- ConstraintLayout defines the relative positions of one view relative to another (above, below, beside, relative to parent) in a device-independent way.

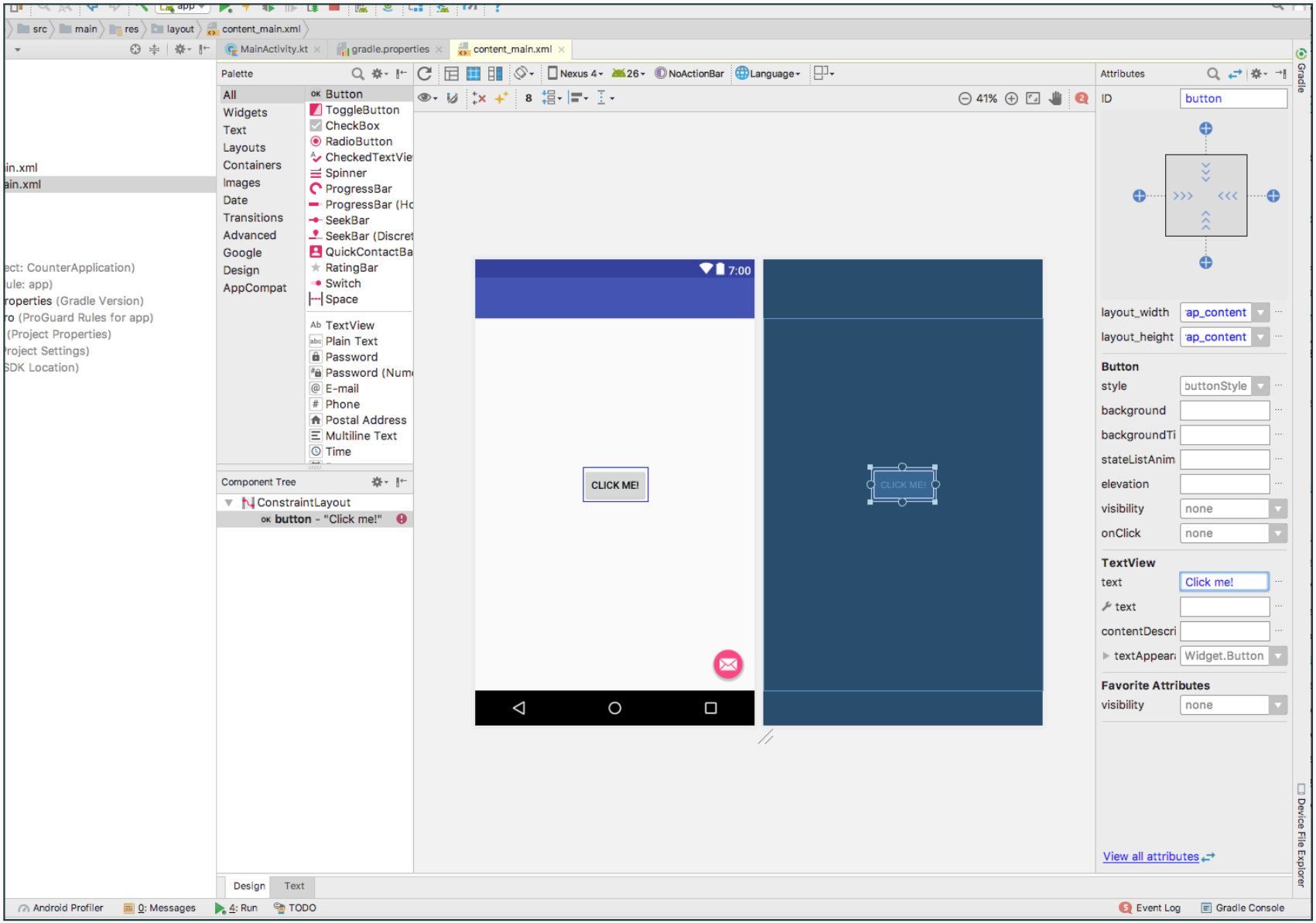
Editing ConstraintLayouts in the Layout Editor



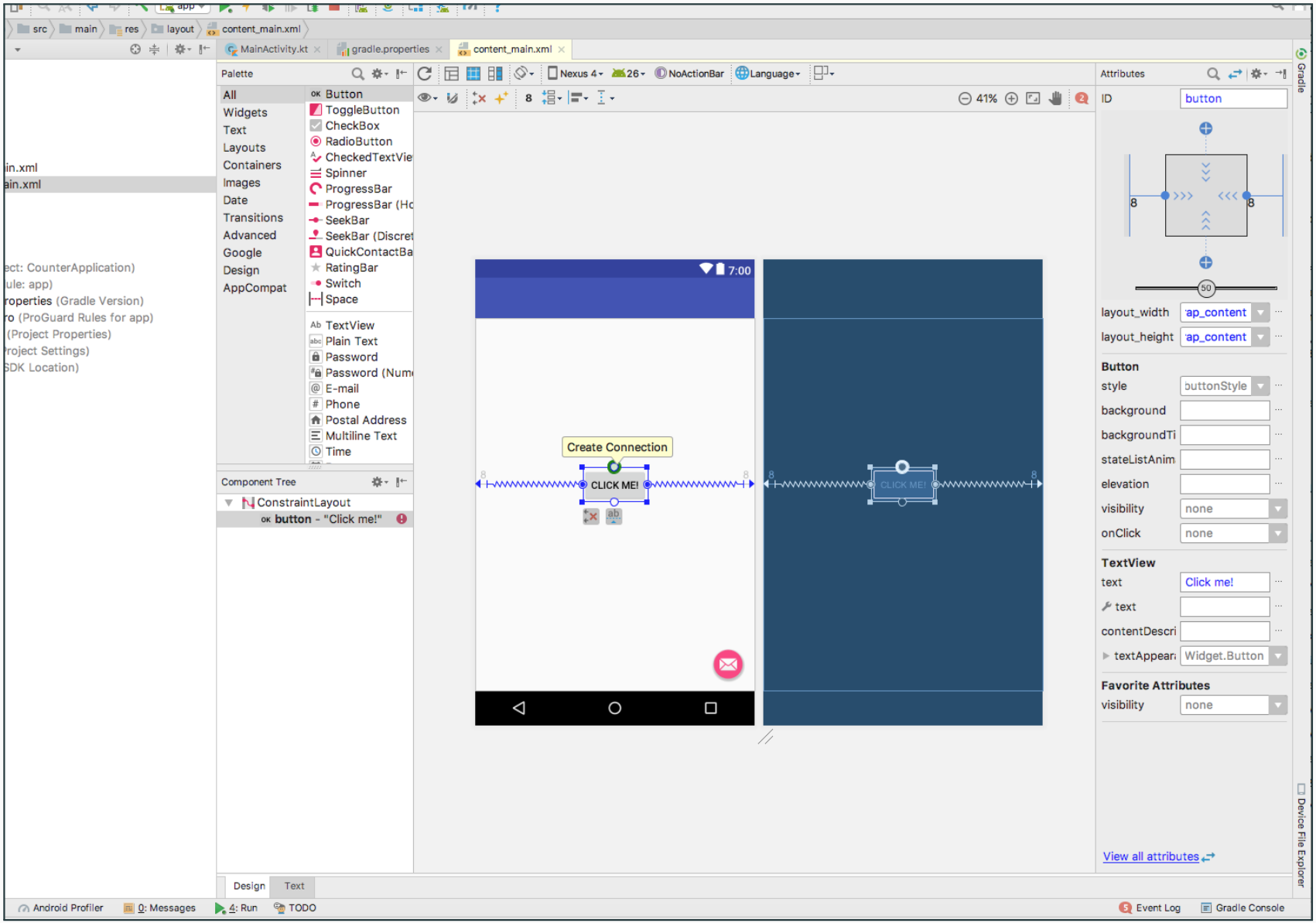
Editing ConstraintLayouts in the Layout Editor



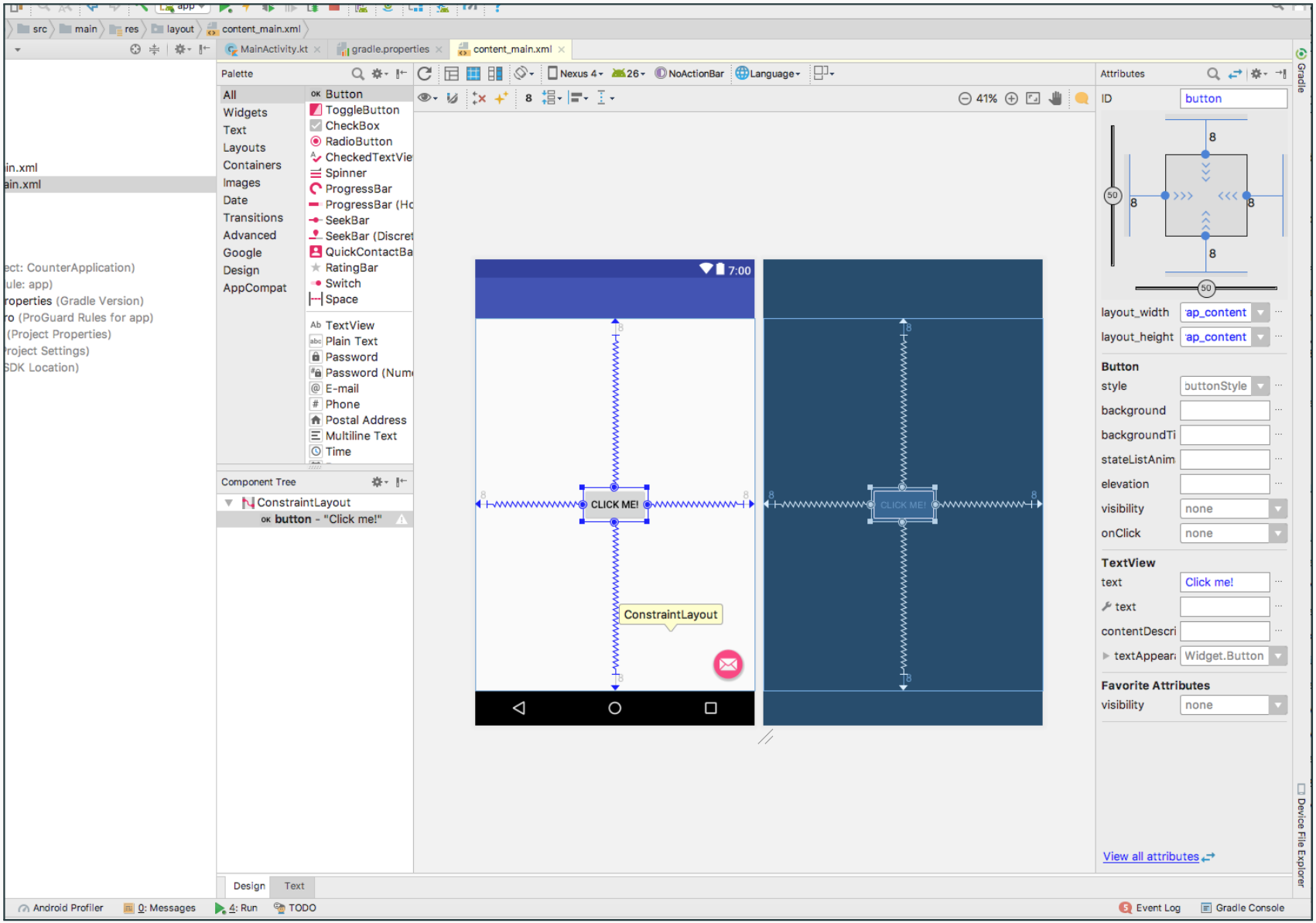
Editing ConstraintLayouts in the Layout Editor



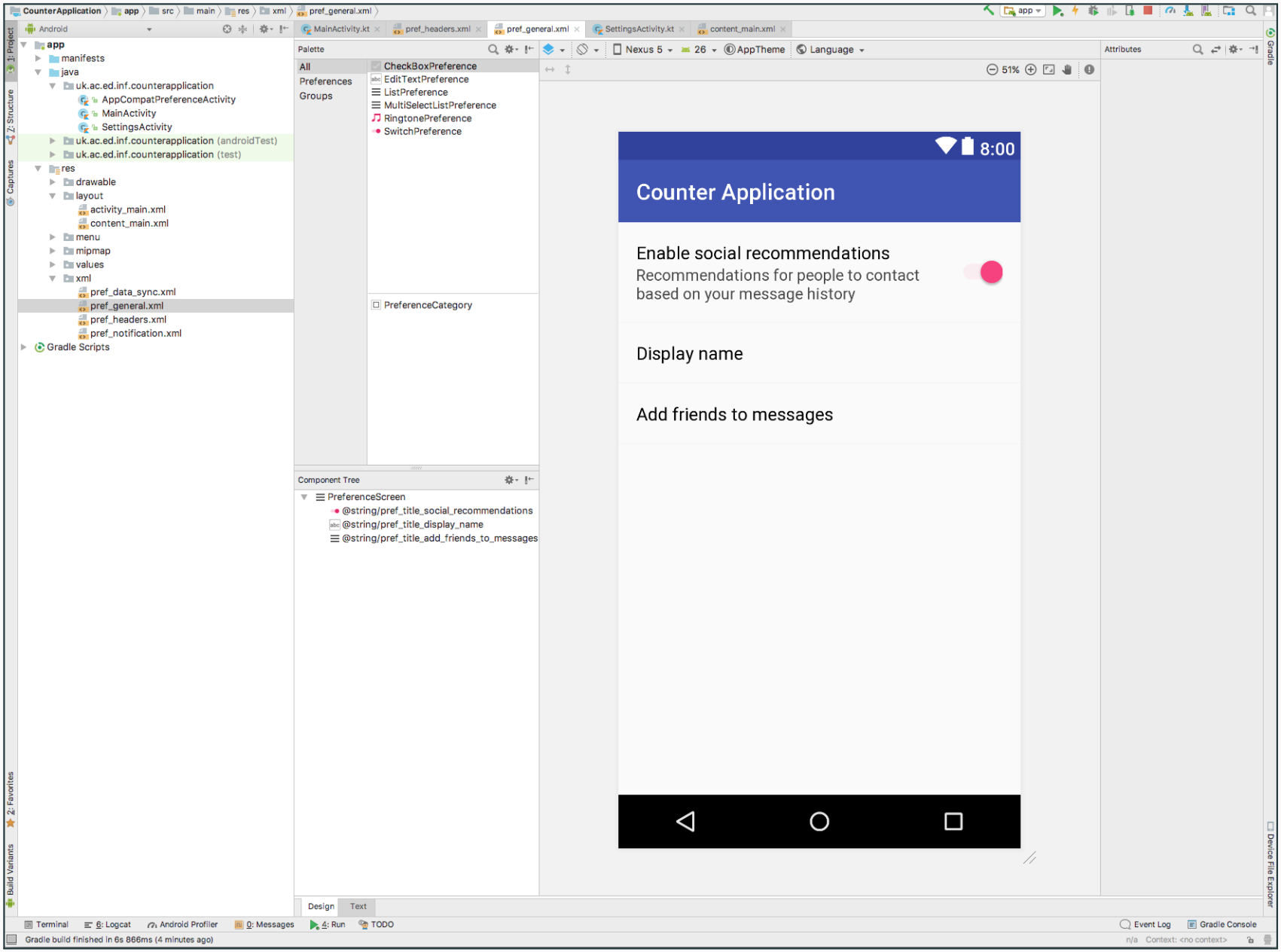
Editing ConstraintLayouts in the Layout Editor



Editing ConstraintLayouts in the Layout Editor



Settings screens can also be edited in the Layout Editor



Links

- https://material.io/guidelines/patterns/settings.html
- https://developer.android.com/guide/topics/ui/settings.html

- https://developer.android.com/training/constraint-layout/index.html
- https://developer.android.com/studio/write/layout-editor.html
- https://medium.com/google-developers/building-interfaces-with-constraintlayout-3958fa38a9f7