Notifying the User
Toasts, Snackbar & Notifications

Sensor Based Mobile Applications

Patrick Ausderau, Ulla Seferlöf, Jarkko Vuori

Helsinki Metropolia University of Applied Science

2022

Outline

Toast

Snackbar

Notification

Lab

For more info:

https://developer.android.com/guide/topics/ui/notifiers/toasts.html

https://developer.android.com/training/snackbar/index.html

https://developer.android.com/reference/kotlin/androidx/compose/material/package-summary#Snackbar(androidx.compose.ui.Modifier,kotlin.Function0,kotlin.Boolean,androidx.compose.ui.graphics.Shape,androidx.compose.ui.graphics.Color,androidx.compose.ui.graphics.Color,androidx.compose.ui.unit.Dp,kotlin.Function0)

https://developer.android.com/guide/topics/ui/notifiers/notifications.html

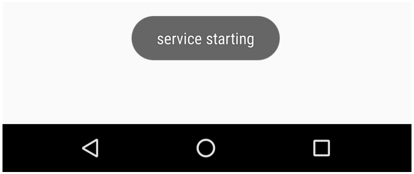
Toast

- Toast provide a small popup message to the user that show at the bottom of device
- will show even if the application is in the background
- fills the amount of space required for the message
- current activity remains visible and interactive when message shows up
- automatically disappear after a timeout
- can be launched by any component

Toast

- if many toasts at the same time, they will show on top of each other
- can be disabled from device settings
- if you need guarantee that the user sees the message, **do not** use Toast (if screen is off or the user is not looking at it, s/he will miss the message)
- no possible interaction, use Snackbar or Notification if you need user action

Toast



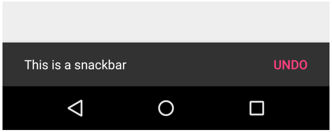
```
// E.g. in the context of an activity
Toast.makeText(
    this,
    R.string.service_start,
    Toast.LENGTH_LONG
).show()

// In a @Composable
val context = LocalContext.current
```

Snackbar

- Snackbar show a small message at bottom of the application
- will only show when the application is active
- if many snackbars at the same time, will be queued to show one by one
- automatically disappear after a timeout or after user interaction (swipe)
- can have user interaction
  - as the Snackbar will disappear after timeout/swipe, your app should provide an alternative way to perform the action
- must be attached to a view (e.g. launched from a `@Composable`, or in traditional XML layout app from an Activity or Fragment)

Snackbar - “Traditional” XML layout



```
// import com.google.android.material.snackbar.Snackbar
// findById coordinator
// somewhere (e.g. in button click)
Snackbar.make(
    coordinator,
    R.string.snackbar_msg,
    Snackbar.LENGTH_LONG
    )
    .setAction(R.string.undo) { Log.d(TAG, "onClick Action...") }
    .show()
```

Note: require a `<androidx.coordinatorlayout.widget.CoordinatorLayout>` (add `@+id/coordinator` (or alike) to it) and in the gradle build: `implementation 'com.google.android.material:material:1.4.0'` Both are done if you use “Basic Activity” when you create your project in Android studio (just add the id to the Coordinator).

Snackbar - @Composable ScaffoldState

```kotlin
@Composable
fun MainView() {
  val scaffoldState = rememberScaffoldState()
  val scope = rememberCoroutineScope()
  val context = LocalContext.current
  Scaffold( scaffoldState = scaffoldState ) {
    // somewhere (e.g. in button click)
    scope.launch {
      scaffoldState.snackbarHostState.showSnackbar(
        context.getString(R.string.snackbar_msg),
        context.getString(R.string.undo),
        SnackbarDuration.Long
      ).let {
        when (it) {
          SnackbarResult.ActionPerformed -> Log.d(TAG, "onClick
              Action...")
          SnackbarResult.Dismissed -> Log.d(TAG, "dismissed...")
        }
      }
    }
  }
}
```

Snackbar - @Composable Snackbar

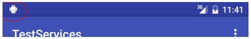
```kotlin
// import androidx.compose.material.Snackbar
// control Snackbar yourself (good idea?)
@Composable
fun ShowSnackbarButton() {
  var showSnack by remember { mutableStateOf(false) }
  Button({ showSnack = !showSnack }) {
    Text(stringResource(if (showSnack) R.string.hide else
        R.string.show))
  }
  if (showSnack) {
    Snackbar(action = {
      TextButton({ Log.d(TAG, "onClick Action...") }) {
        Text(stringResource(R.string.undo))
      }
    }) {
      Text(stringResource(R.string.snackbar_msg))
    }
  }
}
```

Metropolia

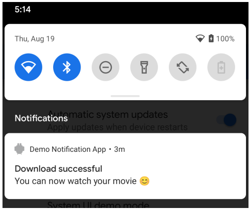
Notification

A notification is a message you can display to the user outside of your application's normal UI

- an icon in the **notification area**



- details available by opening the **notification drawer**


5:14
Thu, Aug 19
100%
Notifications
Automatic system updates
Apply updates when device restarts
Demo Notification App • 3m

Download successful
You can now watch your movie 😊

Note: because the notification runs outside your application, it should follow the specific design guide `https://material.io/design/platform-guidance/android-notifications.html`

Notification

- A `Notification`[1] object should contain at least the following
  - a small icon – `setSmallIcon()` – Required
  - a title – `setContentTitle()`
  - Detail text – `setContentText()`
  - Priority[2] – `setPriority()` – Required for Android $>=8$
- Then create it by calling the `Notification.Builder().build()`
- Finally, use `NotificationManager.notify()` to show the notification in notification area

[1] For legacy projects, make sure you have in your gradle dependencies: `implementation 'com.android.support:support-compat:28.0.0'`
For recent project, it's part of `androidx.core.app.*`, so nothing to add into gradle

[2] If target android $>=8$, set the channel importance: `https://developer.android.com/training/notify-user/build-notification#Priority`

Example

```kotlin
// at top level (before class declaration)
const val CHANNEL_ID = "..."

// when you want to send the notification, e.g. in Compose
val context = LocalContext.current
val notify = NotificationCompat.Builder(context, CHANNEL_ID)
    .setSmallIcon(R.drawable.ic_notify)
    .setContentTitle(stringResource(R.string.notify_title))
    .setContentText("The long description text...")
    .setPriority(NotificationCompat.PRIORITY_DEFAULT)
    .build()

NotificationManagerCompat.from(context).notify(123, notify)
```

Note: to import icon in android studio: File → New → Image Asset

Example

For android $>=$ 8, you must register your notification channel

```kotlin
// e.g. in Activity and call it in onCreate()
private fun createNotificationChannel() {
  // Create the NotificationChannel, but only on API 26+ because
      the NotificationChannel class is new and not in the support
      library
  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    val channel = NotificationChannel(
        CHANNEL_ID,
        getString(R.string.channel_name),
        NotificationManager.IMPORTANCE_DEFAULT
    ).apply {
      description = getString(R.string.channel_description)
    }
    // Register the channel with the system
    val notificationManager =
        getSystemService(NOTIFICATION_SERVICE) as
        NotificationManager
    notificationManager.createNotificationChannel(channel)
  }
}
```

Notification Action

- Notification can have an action when the user tap it (typically open a specific `Activity` on your app).

```
val context = LocalContext.current
val intent = Intent(context, MainActivity::class.java)
intent.putExtra(NOTIFICATION, "some return values...")
val pendingIntent = PendingIntent.getActivity(context, 0, intent,
    0)
val notify = NotificationCompat.Builder(context, CHANNEL_ID)
    /* set icon, title,... */
    .setContentIntent(pendingIntent)
    .build()
```

Notification Button

- Notification can have up to 3 action buttons. Do not use them to duplicate the tap action. Consider a `BroadcastReceiver` that run in background to not block the current active application.
  `addAction(icon: Int, title: CharSequence!, intent: PendingIntent!)`[3]

[3] `https://developer.android.com/training/notify-user/build-notification.html?hl=en#Actions`

Lab_w1_d5_Notification

Modify any of your lab app so that it will show

- a `Snackbar`, (e.g. so user know that some text was updated/modified, if a download was successful (or not),...).
  - Style your Snackbar: `https://material.io/components/snackbars/android#theming-snackbars`
- a `Notification` (try to find something "relevant" to notify). Play with priority level to get the phone ring/vibrate. (Optional: add tap intent and/or try action button).