The 4 Main Android Components

Activity, Service, Broadcast Receiver, Content Provider

Maksym Trushevych
Senior Android Engineer

1

Activities

Activities are the entry point for user interactions with your app. Each activity represents a single screen with a user interface.

- Main building block for user interfaces
- Managed through the activity lifecycle (**onCreate**, **onStart**, **onResume**, etc.)
- Can be launched by other apps with intents
- Follows the "one activity, one purpose" principle

Example: Your app's home screen, a settings page, or checkout screen

2

Activities

Code example

Activity

```kotlin
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Initialize UI components
        val button = findViewById<Button>(R.id.startButton)
        button.setOnClickListener {
            // Launch another activity
            val intent = Intent(this, SecondActivity::class.java)
            startActivity(intent)
        }
    }
}
```

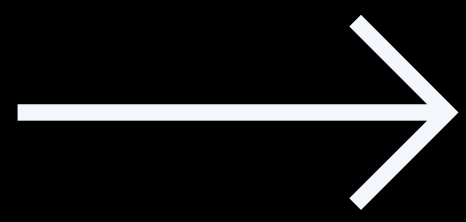
3

Services

Services handle background operations without providing a user interface, allowing your app to perform long-running tasks even when the user isn't actively interacting with it.

- Runs in the background without UI
- Two types: **Started** services and **Bound** services
- Can run even when the app is in the background
- Must be declared in the **AndroidManifest.xml**

Example: Playing music, downloading files, or syncing data



4

Services

Code example

Service

```
class DownloadService : Service() {
    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        // Get URL from intent
        val fileUrl = intent?.getStringExtra("FILE_URL")

        // Start download in background thread
        thread {
            downloadFile(fileUrl)
            stopSelf()
        }

        // If service is killed, restart with pending intent
        return START_REDELIVER_INTENT
    }

    override fun onBind(intent: Intent?): IBinder? = null

    private fun downloadFile(url: String?) {
        // Download implementation
    }
}
```

5 Broadcast Receivers

Broadcast Receivers allow your app to respond to system-wide broadcast messages from the Android system or other apps, even when your app isn't running.

- Responds to system or app events
- Can be registered dynamically or in the manifest
- Executes for a short duration
- Doesn't display a UI but can trigger notifications

Example: Responding to battery low alerts, network changes, or custom events

6

Broadcast Receivers

Code example

Broadcast Receiver

```
class NetworkChangeReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        if (intent.action == ConnectivityManager.CONNECTIVITY_ACTION) {
            val cm = context.getSystemService(Context.CONNECTIVITY_SERVICE) as
ConnectivityManager
            val networkInfo = cm.activeNetworkInfo

            val isConnected = networkInfo != null && networkInfo.isConnected
            if (isConnected) {
                Toast.makeText(context, "Network connected", Toast.LENGTH_SHORT).show()
            } else {
                Toast.makeText(context, "Network disconnected", Toast.LENGTH_SHORT).show()
            }
        }
    }
}

// Register in AndroidManifest.xml or dynamically:
// context.registerReceiver(NetworkChangeReceiver(),
//     IntentFilter(ConnectivityManager.CONNECTIVITY_A
```

7

Content Providers

Content Providers manage shared app data, allowing secure data access and modification between different applications on the device.

- Provides structured data access across apps
- Abstracts underlying storage details
- Uses **URI**-based data addressing
- Supports **CRUD** operations (Create, Read, Update, Delete)

Example: Contacts, calendar events, or media files

8

Content Providers

Code example

ContentProvider

```
class NotesProvider : ContentProvider() {
    companion object {
        private const val AUTHORITY = "com.example.app.provider"
        private const val NOTES_TABLE = "notes"
        val CONTENT_URI: Uri = Uri.parse("content://$AUTHORITY/$NOTES_TABLE")

        // Define columns
        const val _ID = BaseColumns._ID
        const val TITLE = "title"
        const val CONTENT = "content"
    }

    private lateinit var dbHelper: DatabaseHelper

    override fun onCreate(): Boolean {
        dbHelper = DatabaseHelper(context!!)
        return true
    }

    override fun query(uri: Uri, projection: Array<String>?, selection: String?,
                       selectionArgs: Array<String>?, sortOrder: String?): Cursor? {
        val db = dbHelper.readableDatabase
        return db.query(NOTES_TABLE, projection, selection,
                        selectionArgs, null, null, sortOrder)
    }

    // Implement insert(), update(), delete(), getType()...
}

// Using the content provider from another app:
// val cursor = contentResolver.query(NotesProvider.CONTENT_URI, null, null, null, null)
```

Want to learn more?

Follow for more info

Maksym Trushevych

Senior Android Engineer