



Lecture 04: Async Task & List View

IN721: Mobile Application Development

Semester One, 2020

Kaiako: Grayson Orr

Te Kura Matatini ki Otago, Ōtepoti, Aotearoa

Friday, 28 February

LECTURE 03: USER INTERFACES & MATERIAL DESIGN TOPICS

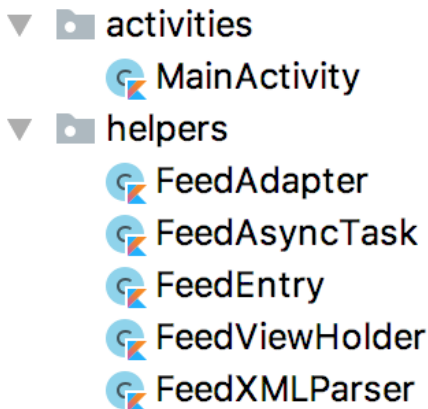
- ▶ Basic activity
- ▶ BottomNavigationView
- ▶ Menus
- ▶ Splash screen
- ▶ Material design

LECTURE 04: ASYNC TASK & LIST VIEW

- ▶ XML parser
- ▶ Permissions
- ▶ Async task
- ▶ List view
- ▶ Custom adapter
- ▶ View holder
- ▶ Saved instance state

APPLICATION STRUCTURE

- ▶ Today's practical application structure
- ▶ You will be provided the MainActivity.kt, FeedXMLParser.kt & layouts



XML Parser: RSS Feed

- ▶ Used by news sites & blogs - provides an XML feed
- ▶ Uploading & parsing XML data is a common task for network-connected apps, for example, LinkedIn
- ▶ Parsing Apple RSS feed

This XML file does not appear to have any style information associated with it. The document tree is shown below.

[illegible]

XML PARSER: ENTRY TAG

- ▶ FeedEntry.kt
- ▶ Name & release date of an free/paid application from iTunes Store

```
class FeedEntry {  
    var name: String = ""  
    var releaseDate: String = ""  
}
```

XML PARSER: CLASS EXAMPLE

- ▶ FeedXMLParser.kt
- ▶ Only class in this course where you will be using asynchronously downloading XML

```
import org.xmlpull.v1.XmlPullParser
import org.xmlpull.v1.XmlPullParserFactory

class FeedXMLParser {
    var feedEntries = ArrayList<FeedEntry>()

    fun parse(data: String): Boolean {
        var isValid = true
        var isInEntryTag = false
        var txtVal = ""

        try {
            val xmlPPFactory: XmlPullParserFactory = XmlPullParserFactory.newInstance()
            xmlPPFactory.isNamespacedAware = true
            val newPullParser: XmlPullParser = xmlPPFactory.newPullParser()
            newPullParser.setInput(data, reader())
            var eventType: Int = newPullParser.eventType
            var feedEntry = FeedEntry()
            while (eventType != XmlPullParser.END_DOCUMENT) {
                val tagName: String? = newPullParser.name?.toLowerCase()
                when (eventType) {
                    XmlPullParser.START_TAG -> if (tagName == "entry") isInEntryTag = true
                    XmlPullParser.TEXT -> txtVal = newPullParser.text
                    XmlPullParser.END_TAG -> {
                        if (isInEntryTag) {
                            when (tagName) {
                                "entry" -> {
                                    feedEntries.add(feedEntry)
                                    isInEntryTag = false
                                    feedEntry = FeedEntry()
                                }
                                "name" -> feedEntry.name = txtVal
                                "releasedate" -> feedEntry.releaseDate = txtVal
                            }
                        }
                    }
                }
                eventType = newPullParser.next()
            }
        } catch (e: Exception) {
            e.printStackTrace()
            isValid = false
        }
        return isValid
    }
}
```

PERMISSIONS

- Indicates whether the app intends to use cleartext network traffic, for example, HTTP

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="graysono.com.cp04asynctasklistview">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">
        <activity android:name=".activities.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```


PERMISSIONS

- ▶ uses-permission tag in the AndroidManifest.xml
- ▶ Internet permission
- ▶ Don't need to ask the user for permission, unlike camera & fine/coarse location

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="graysono.com.cp04asynctasklistview">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">
        <activity android:name=".activities.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>

                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

ASYNCTASK

- ▶ onPostExecute
- ▶ doInBackground
 - ▶ varargs - variable number of arguments
 - ▶ We can pass n number of parameters to a vararg variable of the defined datatype or even of a generic type
- ▶ downloadXML
 - ▶ Gets the entire content of this file as a string using utf-8 or specified charset

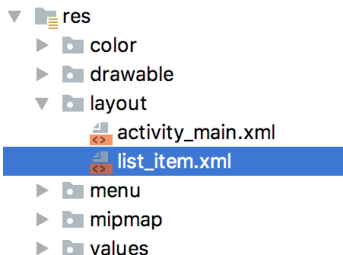
```
class FeedAsyncTask(private var context: Context, private var listView: ListView)
: AsyncTask<String, Void, String>() {
    override fun onPostExecute(result: String) {
        super.onPostExecute(result)
        val feedXMLParser = FeedXMLParser()
        feedXMLParser.parse(result)
        val feedAdapter = FeedAdapter(context, R.layout.list_item, feedXMLParser.feedEntries)
        listView.adapter = feedAdapter
    }

    override fun doInBackground(vararg url: String?): String {
        return try {
            downloadXML(url[0])
        } catch (e: Exception) {
            e.printStackTrace().toString()
        }
    }

    private fun downloadXML(urlPath: String?): String {
        return URL(urlPath).readText()
    }
}
```

LIST VIEW

- ▶ Displays a vertically-scrollable collection of views
- ▶ Legacy, but not deprecated
- ▶ Does not know the details (types & contents) of the views it contains
- ▶ Instead, requests views on demand from a Adapter as needed
- ▶ For a more modern, flexible & performant approach, use a RecyclerView
- ▶ Create a new layout file called list_item.xml



LIST VIEW: XML

► XML - constraint layout & two text views

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

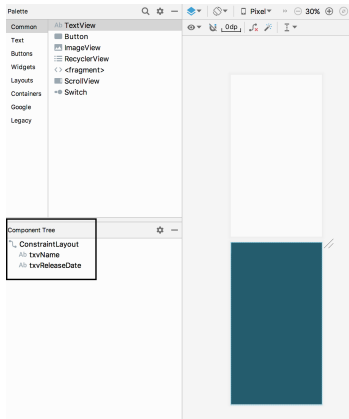
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txvName" app:layout_constraintTop_toTopOf="parent"
        android:textSize="18sp"
        android:textStyle="bold" app:layout_constraintStart_toStartOf="parent"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/txvReleaseDate"
        android:textSize="14sp"
        app:layout_constraintTop_toBottomOf="@+id/txvName"
        app:layout_constraintStart_toStartOf="parent" android:layout_marginTop="4dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

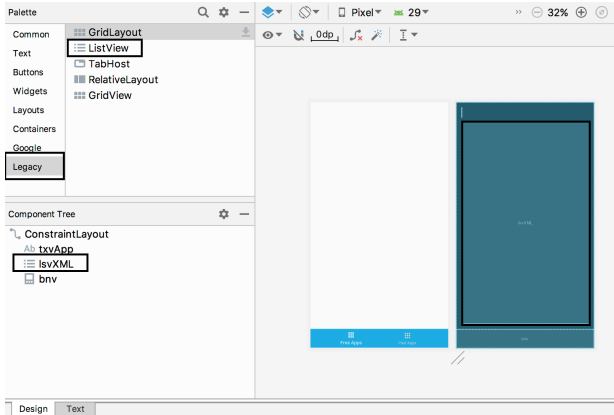
LISTVIEW

- ▶ Next week, we will be using a text view & image view to display artist information from the Last.fm API



LISTVIEW

- ▶ Add a list view widget to activity_main.xml
- ▶ Under legacy tab in the widget palette
- ▶ You should see an empty list view widget



CUSTOM ADAPTER

- ▶ Last time, we use an in-built ArrayAdapter provided by the Android system
- ▶ getCount - returns the number of items in the ArrayAdapter
- ▶ getView - returns a view that displays the data at the specified position in the data set

```
class FeedAdapter(context: Context, private var resource: Int,
                  private var feedEntries: ArrayList<FeedEntry>) :
    ArrayAdapter<FeedEntry>(context, resource) {
    override fun getCount(): Int {
        return if (feedEntries.isNotEmpty()) feedEntries.size else 0
    }

    override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
        val view: View
        val feedViewHolder: FeedViewHolder
        if (convertView == null) {
            view = LayoutInflater.from(context).inflate(resource, parent, attachToRoot: false)
            feedViewHolder = FeedViewHolder(view)
            view.tag = feedViewHolder
        } else {
            view = convertView
            feedViewHolder = view.tag as FeedViewHolder
        }

        val feedEntry: FeedEntry = feedEntries[position]
        val releaseDate: String = feedEntry.releaseDate.substring(0, 10)

        feedViewHolder.tvName.text = feedEntry.name
        feedViewHolder.tvReleaseDate.text = "Release Date: $releaseDate"

        return view
    }
}
```

VIEW HOLDER

- ▶ Make sure you adhere to this pattern
- ▶ Increases the speed when rendering data - this also applies to RecyclerView
- ▶ The number of times which the findViewById method is invoked is reduced
- ▶ Existing views don't have to be garbage collected & new views do not have to be inflated

```
class FeedViewHolder(var view: View) {  
    var txtvName: TextView = view.findViewById(R.id.txtvName)  
    var txtvReleaseDate: TextView = view.findViewById(R.id.txtvReleaseDate)  
}
```


SAVED INSTANCE STATE

- ▶ The user will expect that when they start an activity, the user interface state will remain the same until the user completely dismisses the activity
- ▶ Not the case when changing the orientation from portrait to landscape - destroys then creates the activity
- ▶ onSaveInstanceState

```
override fun onSaveInstanceState(outState: Bundle) {  
    super.onSaveInstanceState(outState)  
    outState.putString(feedUrlKey, feedUrl)  
    outState.putString(feedTitleKey, feedTitle)  
}
```

SAVED INSTANCE STATE

- ▶ In the onCreate in MainActivity.kt
- ▶ Check if the save instance state is not null or contains at least one key/value
- ▶ Get the value from the save instance state using the key
- ▶ Ignore the warnings...sometimes we don't need to worry about them

```
if (savedInstanceState != null) {  
    feedUrl = savedInstanceState.getString(feedUrlKey)  
    feedTitle = savedInstanceState.getString(feedTitleKey)  
}
```

PRACTICAL

- ▶ Series of tasks covering today's lecture
- ▶ Worth 2% of your final mark for the Design and Development of Applications for Mobile Devices course
- ▶ Deadline: Friday, 12 June at 5pm