

# 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
  - activities
    - 🛖 MainActivity
    - helpers
      - FeedAdapter
      - FeedAsyncTask
      - FeedEntry
      - FeedViewHolder
      - FeedXMLParser

#### 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
v<feed mmlns:im="http://itunes.apple.com/rss" xmlns="http://www.w3.org/2005/Atom" xml:lang="en">
            http://ax.itunes.apple.com/WebChjects/MEStoreGervices.wom/ws/MESK/topsongs/limit=10/xml
         *title*iTunes Store: Top Sonos*/title*
         cupdated>2020-02-27855148132-07100</updated>
         clink rel"alterant type"text/html href"https://ivase.apple.com/MedDjects/MEStore.vom/va/viewtoptc=usaid=1spoptd=1"/>
clink rel"alt" bref"http://ax.items.apple.com/MeDDjects/MEStoreServices.vom/va/MEStopomys/limit=10/mml"/>
            <nome>iTemes Store</page)</pre>
              <url>http://www.apple.com/itumes/</uri>
         <rights>Copyright 2018 Apple Inc.</rights>
              entry:
<updated>2010-01-27705:49:32-07:00</updated>
                  https://masic.ample.com/ss/album/the-other-side/148757453871=14875745374so=2
              cuitleoThe Other Side - SIA & Justin Timberlakec/titleo
          vanamerms other Bides/immame/
«link rel="alternate" type="text/btml" href="https://masic.apple.com/us/album/the-other-mide/148757453671=14875745374co=2"/>
vincenteerbyse term="masic" label="masic">
vincenteerbyse term="masic" label="masic" label="masic">
vincenteerbyse term="masic" label="masic" label="masic"
vincenteerbyse term="masic" label="masic" l
                  <im:contentType term="Track" label="Track"/>
              trateoury inside 14' term="Pop" achese="https://music.apole.com/us/osare/music-oco/idi47xo=2" label="Pop"/>
          - Clastic tipe Teview rel realisance type "ministrative translation of the Company of the Compan
              <invariat href="https://mosic.opple.com/us/artist/sca/6038003347so=2">SEA 4 Justin Timberlake</invartist>
                <im:price amount='1,29800' currency='USD'+51,29*/im:price>
                    https://isl.ms.in.ponahio.com/image/thumh/Masie113/vd/e#/mb/he/eM/mb/hb/3:21h/:/52amil/he/ff/MB646f/RB648132892.ims/NiwhNhh.ms
         */incinage/
                    mmingo Maight-40>
https://id-asi.msetatio.com/image/thumb/Masioll3/v4/e9/b0/9e/e9h09e53-21b7-52Ba-bb5e-7758810be66f/886448132882.jpq/60x60bb.png
                  https://id-ed.puratic.com/image/thumb/tunic113/v4/e9/h0/ee/e9h0/e53-21h7-524a-bb5e-7758810be66f/886488132892.ipg/178x178bb.po
                  9 This Compilation (F) 2020 NCA Records, a division of Sony Music Entertainment
              <in:releaseDete label='February 26, 2020'>2020-02-26700:00:00-07:00
```

#### 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.isNamespaceAware = 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
                    Xm1Pul1Parser.FND TAG -> {
                        if (isInEntryTag) {
                            when (tagName)
                                    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"</pre>
          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:supportsRtl="true"
            android:usesCleartextTraffic="true
            <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

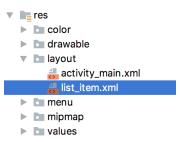
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"</pre>
          package="graysono.com.cp04asynctasklistview">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
            android:allowBackup="true"
            android:icon="@mipmap/ic launcher"
            android:label="@string/app name"
            android:roundIcon="@mipmap/ic launcher round"
            android:supportsRtl="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>
```

## ASYNC TASK

- ▶ onPostExecute
- dolnBackground
  - 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

#### 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



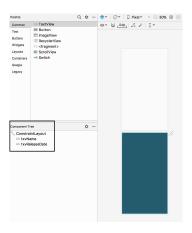
#### LIST VIEW: XML

XML - constraint layout & two text views

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout</pre>
        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>
```

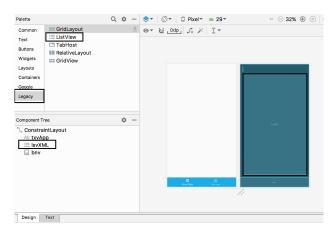
### LISTVIEW

► 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.txvName.text = feedEntry.name
        feedViewHolder.txvReleaseDate.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 findViewByld 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 <u>view</u>: View) {
    var <u>txvName</u>: TextView = <u>view</u>.findViewById(R.id.txvName)
    var <u>txvReleaseDate</u>: TextView = <u>view</u>.findViewById(R.id.txvReleaseDate)
}
```

#### 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
- onSavedInstanceState

```
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