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

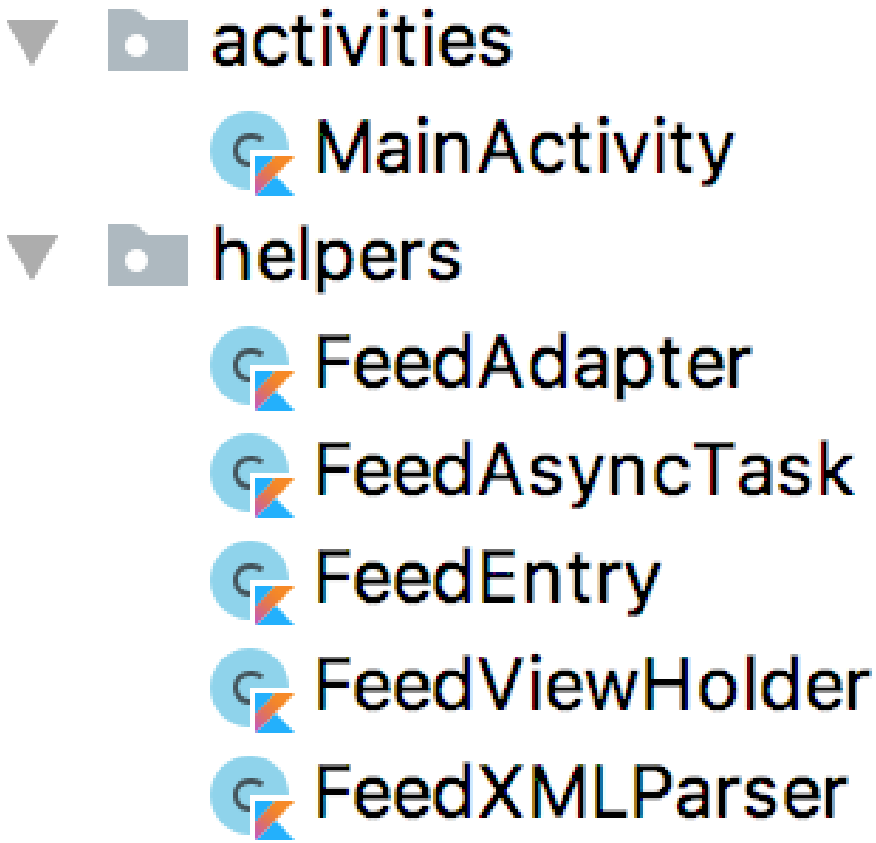
- Basic activity
- BottomNavigationView
- Menus
- Splash screen
- Material design

Lecture 04: Async Task & List View Topics

- 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.


```
▼<feed xmlns:im="http://itunes.apple.com/rss" xmlns="http://www.w3.org/2005/Atom" xml:lang="en">
  ▼<id>
     http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/topsongs/limit=10/xml
   </id>
   <title>iTunes Store: Top Songs</title>
   <updated>2020-02-27T05:48:32-07:00</updated>
   <link rel="alternate" type="text/html" href="https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewTop?cc=us&id=1&popId=1"/>
   <link rel="self" href="http://ax.itunes.apple.com/WebObjects/MZStoreServices.woa/ws/RSS/topsongs/limit=10/xml"/>
   <icon>http://itunes.apple.com/favicon.ico</icon>
  ▼<author>
     <name>iTunes Store</name>
     <uri>http://www.apple.com/itunes/</uri>
   </author>
   <rights>Copyright 2008 Apple Inc.</rights>
  ▼<entry>
     <updated>2020-02-27T05:48:32-07:00</updated>
    ▼<id im:id="1487574537">
       https://music.apple.com/us/album/the-other-side/1487574536?i=1487574537&uo=2
     </id>
     <title>The Other Side - SZA & Justin Timberlake</title>
     <im:name>The Other Side</im:name>
     <link rel="alternate" type="text/html" href="https://music.apple.com/us/album/the-other-side/1487574536?i=1487574537&uo=2"/>
    ▼<im:contentType term="Music" label="Music">
       <im:contentType term="Track" label="Track"/>
     </im:contentType>
     <category im:id="14" term="Pop" scheme="https://music.apple.com/us/genre/music-pop/id14?uo=2" label="Pop"/>
    ▼<link title="Preview" rel="enclosure" type="audio/x-m4a" href="https://audio-ssl.itunes.apple.com/itunes-assets/AudioPreview123/v4/59/89/f4/5989f44a-fbab-4fef-dbf3-
     1b0f2d891c3a/mzaf_2131618393908239776.plus.aac.p.m4a" im:assetType="preview">
       <im:duration>30000</im:duration>
     </link>
     <im:artist href="https://music.apple.com/us/artist/sza/605800394?uo=2">SZA & Justin Timberlake</im:artist>
     <im:price amount="1.29000" currency="USD">$1.29</im:price>
    ▼<im:image height="55">
       https://is2-ssl.mzstatic.com/image/thumb/Music113/v4/e9/b0/9e/e9b09e53-21b7-520a-bb5e-7758810be66f/886448132882.jpg/55x55bb.png
     </im:image>
    ▼<im:image height="60">
       https://is4-ssl.mzstatic.com/image/thumb/Music113/v4/e9/b0/9e/e9b09e53-21b7-520a-bb5e-7758810be66f/886448132882.jpg/60x60bb.png
     </im:image>
    ▼<im:image height="170">
       https://is4-ssl.mzstatic.com/image/thumb/Music113/v4/e9/b0/9e/e9b09e53-21b7-520a-bb5e-7758810be66f/886448132882.jpg/170x170bb.png
     </im:image>
    ▼<rights>
       ℗ This Compilation (P) 2020 RCA Records, a division of Sony Music Entertainment
     </rights>
     <im:releaseDate label="February 26, 2020">2020-02-26T00:00:00-07:00</im:releaseDate>
```

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
                    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: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>
```

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: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>
```

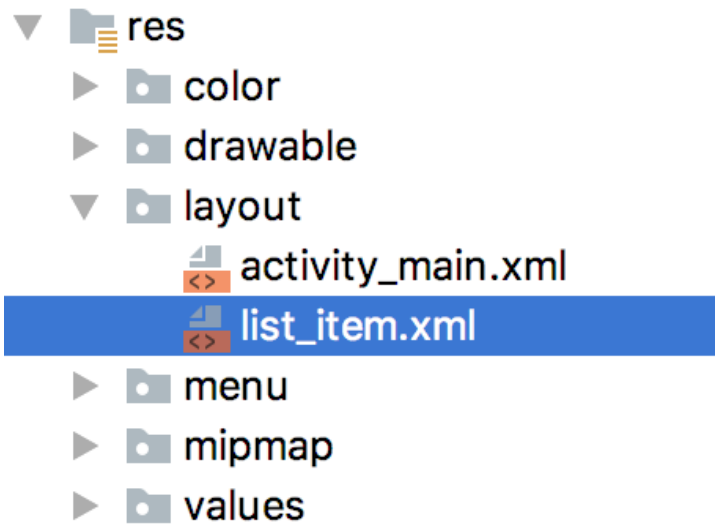
Async Task

- onPostExecute
- 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



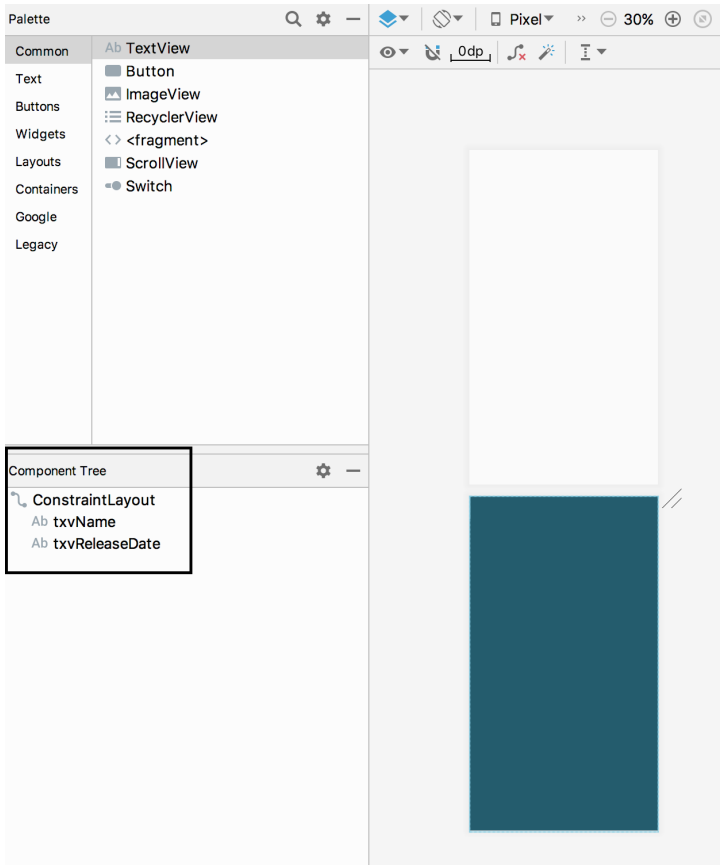
List View: XML

- XML - constraint layout & two text views

```xml
<?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>
```

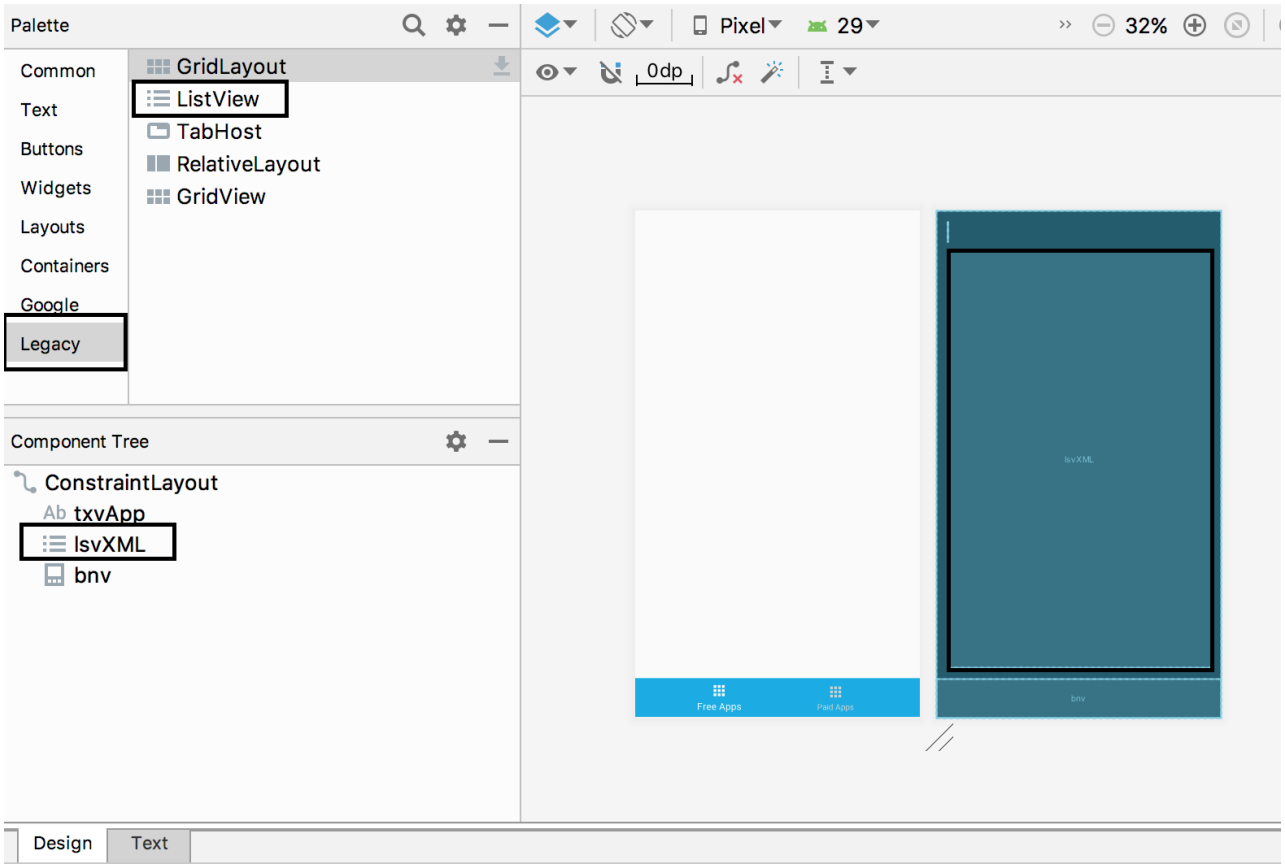
ListView

- 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.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 findViewById method is invoked is reduced
- Existing views donť have to be garbage collected & new views do not have to be inflated

```
class FeedViewHolder(var view: View) {
    var txvName: TextView = view.findViewById(R.id.txvName)
    var txvReleaseDate: TextView = view.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