



Lecture 05: Async Task & Recycler View

IN721: Design and Development of Applications for Mobile Devices

Semester One, 2020

Kaiako: Grayson Orr

Te Kura Matatini ki Otago, Ōtepoti, Aotearoa

Wednesday, 4 March

LECTURE 04: ASYNC TASK & LIST VIEW TOPICS

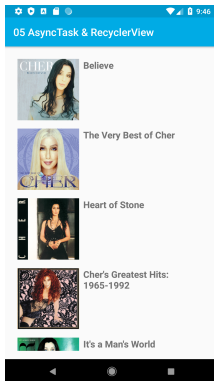
- ▶ XML parser
- ▶ Permissions
- ▶ Async task
- ▶ List view
- ▶ Custom adapter
- ▶ View holder
- ▶ Saved instance state

LECTURE 05: ASYNC TASK & RECYCLER VIEW

- ▶ Picasso
- ▶ Custom adapter - recycler view
- ▶ Async task - JSON
- ▶ Enum
- ▶ Interface
- ▶ Base activity

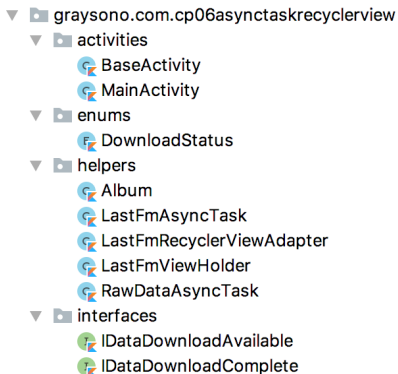
APPLICATION SCREENSHOT

- Simple application using Last.fm API



FILE STRUCTURE

- ▶ Two activities
- ▶ One enumeration
- ▶ Five helpers
- ▶ Two interfaces



ALBUM

- ▶ Album object - data helper class
 - ▶ name: String
 - ▶ playCount: Int
 - ▶ artist: String
 - ▶ image: String

```
data class Album(  
    var name: String,  
    var playCount: Int,  
    var artist: String,  
    var image: String  
)
```

ENUM

- ▶ DownloadStatus.kt
 - ▶ Check for no response - default
 - ▶ Check for ok response - fetching data from a web service
 - ▶ Should be checking for different exceptions - you should be thinking about this
 - ▶ Types of checks would be made in the async task class
 - ▶ Two checks are sufficient at the moment

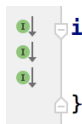
```
enum class DownloadStatus {  
    NONE,  
    OK  
}
```

INTERFACE

- ▶ `IDataDownloadAvailable.kt`
- ▶ `IDataDownloadComplete.kt`

INTERFACE - IDataDownloadAvailable.KT

- ▶ onDataAvailable(ArrayList<Album>)
 - ▶ When implemented, loads the available data from the Album array list into the recycler view
- ▶ onError(Exception)
 - ▶ Raised if there is an error loading the available data from the Album array list



```
interface IDataDownloadAvailable {  
    fun onDataAvailable(data: ArrayList<Album>)  
    fun onError(e: Exception)  
}
```

INTERFACE - IDataDownloadComplete.KT

- ▶ onDownloadComplete(String, DownloadStatus)
 - ▶ When implemented, gets the status (enum value) of the async task - checks if its completed

```
interface IDataDownloadComplete {  
    fun onDownloadComplete(data: String, status: DownloadStatus)
```

PICASSO

- ▶ Android Picasso
- ▶ An image loading/processing library developed & maintained by Square Inc
- ▶ Simplifies the process of displaying images from external locations
- ▶ Add the following dependency in your **build.gradle** file

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"  
    implementation 'com.android.support:appcompat-v7:28.0.0'  
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
    implementation 'com.android.support:design:28.0.0'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.2'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
    implementation 'com.android.support:recyclerview-v7:28.0.0'  
    implementation 'com.squareup.picasso:picasso:2.5.2'
```

CREATE URI - BUILDER PATTERN

- ▶ Students taking OOSD will learn more about this pattern in the coming weeks
- ▶ Chaining query parameters to the base url
- ▶ Convert the query into a query string

```
private fun createURI(  
    baseUrl: String, method: String, artist: String,  
    apiKey: String, format: String  
): String {  
    return Uri.parse(baseUrl)  
        .buildUpon()  
        .appendQueryParameter("method", method)  
        .appendQueryParameter("artist", artist)  
        .appendQueryParameter("api_key", apiKey)  
        .appendQueryParameter("format", format)  
        .build().toString()  
}
```

CUSTOM ADAPTER - RECYCLER VIEW

- ▶ Slightly different to the custom adapter we saw last session
- ▶ onBindViewHolder
- ▶ onCreateViewHolder

```
class LastFmRecyclerViewAdapter(private var albums: ArrayList<Album>) :  
    RecyclerView.Adapter<LastFmViewHolder>() {  
    override fun getItemCount(): Int {  
        return if (albums.isNotEmpty()) albums.size else 0  
    }  
  
    fun loadNewData(newAlbums: ArrayList<Album>) {  
        albums = newAlbums  
        notifyDataSetChanged() // Observable method call  
    }  
  
    override fun onBindViewHolder(viewHolder: LastFmViewHolder, position: Int) {  
        val album: Album = albums[position] // Get position of album in the array list  
        Picasso.with(viewHolder.itemView.context)  
            .load(album.image) // Load the image from Album  
            .placeholder(R.drawable.ic_image_black_48dp) // Set a default image  
            .error(R.drawable.ic_image_black_48dp) // Set an error image  
            .into(viewHolder.imageView) // Put image into the view holder image view  
  
        viewHolder.textView.text = album.name  
    }  
  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): LastFmViewHolder {  
        val view: View =  
            LayoutInflater.from(parent.context).inflate(R.layout.list_item, parent, attachToRoot: false)  
        return LastFmViewHolder(view)  
    }  
}
```

VIEW HOLDER

- Much the same as we saw last session

```
class LastFmViewHolder(view: View) : RecyclerView.ViewHolder(view) {  
    var txvName: TextView = view.findViewById(R.id.txvName)  
    var imvAlbum: ImageView = view.findViewById(R.id.imvAlbum)  
}
```

ASYNC TASK - RAW DATA

- ▶ Generic async task - open/closed principle
- ▶ XML we saw last time only works on Apple RSS feeds - breaks the open/closed principle
- ▶ In the `doInBackground()`, you could do the following checks:
 - ▶ `MalformedURLException`
 - ▶ `IOException`
 - ▶ `SecurityException`

```
class RawDataAsyncTask(private val listener: IDataDownloadComplete) :  
    AsyncTask<String, Void, String>() {  
    private var downloadStatus = DownloadStatus.NONE  
  
    override fun onPostExecute(result: String) {  
        listener.onDownloadComplete(result, downloadStatus)  
    }  
  
    override fun doInBackground(vararg url: String?): String {  
        var data = ""  
        try {  
            downloadStatus = DownloadStatus.OK  
            data = downloadXML(url[0])  
        } catch (e: Exception) {  
            e.printStackTrace()  
        }  
        return data  
    }  
  
    private fun downloadXML(urlPath: String?): String {  
        return URL(urlPath).readText()  
    }  
}
```

ASYNC TASK - JSON

- ▶ Parsing JSON data from Last.fm API
- ▶ Parsing data works the same in most languages - slight syntax differences & use of libraries
- ▶ In order to access this data, we need an API key

```
class LastFmAsyncTask(private val listener: IDownloadAvailable) :  
    AsyncTask<String, Void, ArrayList<Album>>() {  
    override fun onPostExecute(result: ArrayList<Album>) {  
        super.onPostExecute(result)  
        listener.onDataAvailable(result)  
    }  
  
    override fun doInBackground(vararg url: String?): ArrayList<Album> {  
        val albums = ArrayList<Album>()  
        try {  
            val jsonData = JSONObject(url[0])  
            val topAlbumsObj: JSONObject = jsonData.getJSONObject( name: "topalbums")  
            val albumItems: JSONArray = topAlbumsObj.getJSONArray( name: "album")  
  
            for (albumItem: Int in 0 until albumItems.length()) {  
                val albumObj: JSONObject = albumItems.getJSONObject(albumItem)  
                val name: String = albumObj.getString( name: "name")  
                val playCount: Int = albumObj.getInt( name: "playcount")  
                val artist: JSONObject = albumObj.getJSONObject( name: "artist")  
                val artistName: String = artist.getString( name: "name")  
                val imageItems: JSONArray = albumObj.getJSONArray( name: "image")  
                val imageObj: JSONObject = imageItems.getJSONObject( index: 3)  
                val imageText: String = imageObj.getString( name: "#text")  
                if (imageText.isNotEmpty()) {  
                    val album = Album(name, playCount, artistName, imageText)  
                    albums.add(album)  
                }  
            }  
        } catch (e: Exception) {  
            cancel( mayInterruptIfRunning: true)  
            listener.onError(e)  
        }  
        return albums  
    }  
}
```


LAST.FM API KEY

- ▶ Go to <https://www.last.fm/api/show/artist.getTopAlbums>
- ▶ Click on the JSON example URL - you will also see an XML example URL
- ▶ Note that we don't see any JSON data...why? We don't have an API key or the API key we have is invalid
- ▶ You need to setup an API account to use the API. To save you the hassle, I will let you use mine
 - ▶ Note: this is not best practices. You should ignore your API key when storing your application on a public domain, for example, GitHub, GitLab
- ▶ Here is my API key - 58384a2141a4b9737eacb9d0989b8a8c
- ▶ Replace YOUR_API_KEY with the hash above - you should now see a JSON object

LAST.FM JSON DATA

- ▶ topalbums object
- ▶ album array
 - ▶ album objects within album array
 - ▶ album objects hold information (key/value pair) such as name, play count, url, artist & image

```
{
  - topalbums: {
    - albums: [
      - {
        name: "Believe",
        playcount: 2587971,
        mbid: "43b3a3ca-26f2-4a2b-bd47-647atec2bebd",
        url: "https://www.last.fm/music/Believe",
        - artist: {
          name: "Cher",
          mbid: "bfcc6d75-e6a5-4bc6-8282-47aacc8531818",
          url: "https://www.last.fm/music/Cher"
        },
        - image: [
          - {
            #text: "https://lastfm.freetls.fastly.net/i/u/34a/b2c2311a9a7f9edbc8b945074ca3b.png",
            size: "small"
          },
          - {
            #text: "https://lastfm.freetls.fastly.net/i/u/64a/b2c2311a9a7f9edbc8b945074ca3b.png",
            size: "medium"
          },
          - {
            #text: "https://lastfm.freetls.fastly.net/i/u/174a/b2c2311a9a7f9edbc8b945074ca3b.png",
            size: "large"
          },
          - {
            #text: "https://lastfm.freetls.fastly.net/i/u/380a3012a0c2311a9a7f9edbc8b945074ca3b.png",
            size: "extralarge"
          }
        ]
      },
      - {
        name: "The Very Best of Cher",
        playcount: 1491982,
        mbid: "a7a6d6d7-ae711-4bee-9db1-b31a118beaf5",
        url: "https://www.last.fm/music/The+Very+Best+of+Cher",
        - artist: {
          name: "Cher",
          mbid: "bfcc6d75-e6a5-4bc6-8282-47aacc8531818",
          url: "https://www.last.fm/music/Cher"
        },
        - image: [
          - {
            #text: "https://lastfm.freetls.fastly.net/i/u/34a/287bc1457792451399d8f4d4455e91.png",
            size: "small"
          },
          - {
            #text: "https://lastfm.freetls.fastly.net/i/u/64a/287bc1457792451399d8f4d4455e91.png",
            size: "medium"
          },
          - {
            #text: "https://lastfm.freetls.fastly.net/i/u/174a/287bc1457792451399d8f4d4455e91.png",
            size: "large"
          },
          - {
            #text: "https://lastfm.freetls.fastly.net/i/u/380a3012a0c2311a9a7f9edbc8b945074ca3b.png",
            size: "extralarge"
          }
        ]
      }
    ]
  },
}
```

BASE ACTIVITY

- ▶ BaseActivity.kt - open class
- ▶ MainActivity.kt with implement BaseActivity.kt instead of AppCompatActivity()
- ▶ @SuppressWarnings
 - ▶ import android.annotation.SuppressLint
 - ▶ Indicates that lint (orange highlight) should ignore the specified warnings for the annotated element
- ▶ Enable the back button on the support action bar using an internal function - displayToolbar(boolean)
 - ▶ supportActionBar?.setDisplayHomeAsUpEnabled(boolean) - check for null

```
@SuppressWarnings( ...value: "Registered")
open class BaseActivity: AppCompatActivity() {
    internal fun displayToolbar(enableHome: Boolean) {
        setSupportActionBar(toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(enableHome)
    }
}
```

PRACTICAL

- ▶ Series of tasks covering today's lecture
- ▶ Worth 1% of your final mark for the Design and Development of Applications for Mobile Devices course
- ▶ Deadline: Friday, 12 June at 5pm

EXAM 01

- ▶ Series of tasks covering lectures 01-04
- ▶ Worth 6% of your final mark for the Design and Development of Applications for Mobile Devices course
- ▶ Release: Friday, 6, March at 5pm
- ▶ Deadline: Friday, 13 March at 5pm