



Lecture 06: Card View & Parcelable 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 05: ASYNC TASK & RECYCLER VIEW TOPICS

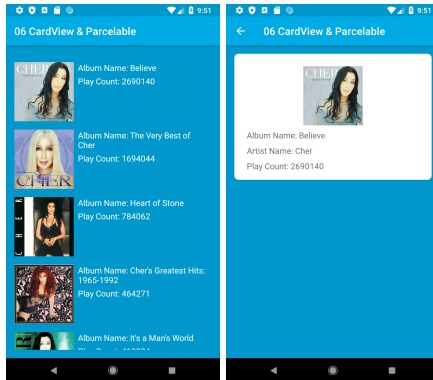
- ▶ File structure
- ▶ Album object
- ▶ Enums & interfaces
- ▶ Picasso
- ▶ Custom adapter - recycler view
- ▶ View holder
- ▶ Async task - JSON
- ▶ API keys
- ▶ Base activity

LECTURE 06: CARD VIEW & PARCELABLE TOPICS

- ▶ File structure
- ▶ Card view
- ▶ Parcelable

APPLICATION SCREENSHOT

- Extension to the simple application using Last.fm API








FILE STRUCTURE

- ▶ Adding new files to our current structure:
 - ▶ DetailsActivity.kt - activities directory
 - ▶ RecyclerViewItemClickListener.kt - helpers directory
 - ▶ IRecyclerViewItem.kt - interfaces directory

DETAILSACTIVITY

- ▶ Create a new class called DetailsActivity.kt
- ▶ An activity & content XML file will be generated for you
- ▶ Remove the floating action button from the activity XML file

▼ layout

-  activity_details.xml
-  activity_main.xml
-  content_details.xml
-  content_main.xml
-  list_item.xml

DETAILSACTIVITY

- DetailsActivity.kt implements BaseActivity.kt

```
class DetailsActivity : BaseActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_details)  
        displayToolbar(isHomeEnabled: true)  
        val album: Album? = intent.extras?.getParcelable(key: "album")  
  
        txvAlbumName.text = "Album Name: {album?.name}"  
        txvAlbumArtist.text = "Artist Name: {album?.artist}"  
        txvAlbumPlayCount.text = "Play Count: {album?.playCount.toString()}"  
  
        Picasso.with(context: this@DetailsActivity).load(album?.image)  
            .error(R.drawable.ic_image_black_48dp)  
            .placeholder(R.drawable.ic_image_black_48dp)  
            .into(imvAlbumImage)  
    }  
}
```

DETAILSACTIVITY

- ▶ Getting the Album's properties in the parcel
- ▶ You will investigate this later as part of your research tasks

```
class DetailsActivity : BaseActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_details)  
        displayHomeAsUpEnabled(true)  
        val album: Album? = intent.extras?.getParcelable(key: "album")  
  
        txvAlbumName.text = "Album Name: {album?.name}"  
        txvAlbumArtist.text = "Artist Name: {album?.artist}"  
        txvAlbumPlayCount.text = "Play Count: {album?.playCount.toString()}"  
  
        Picasso.with(context: this@DetailsActivity).load(album?.image)  
            .error(R.drawable.ic_image_black_48dp)  
            .placeholder(R.drawable.ic_image_black_48dp)  
            .into(imvAlbumImage)  
    }  
}
```


DETAILSACTIVITY

- ▶ Binding each Album property to a text view in the card view widget
- ▶ You will also investigate how to implement a card view widget as part of your research tasks
- ▶ Note: the card view widget will go inside the content XML file
- ▶ In the screenshot, string placeholders are used. This is why it is highlighted with grey text

```
class DetailsActivity : BaseActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_details)  
        displayToolbar( isHomeEnabled: true)  
        val album: Album? = intent.extras.getParcelable( key: "album")  
  
        txvAlbumName.text = "Album Name: {album?.name}"  
        txvAlbumArtist.text = "Artist Name: {album?.artist}"  
        txvAlbumPlayCount.text = "Play Count: {album?.playCount.toString()}"  
  
        Picasso.with( context: this@DetailsActivity).load(album?.image)  
            .error(R.drawable.ic_image_black_48dp)  
            .placeholder(R.drawable.ic_image_black_48dp)  
            .into(imvAlbumImage)  
    }  
}
```

DETAILSACTIVITY

- ▶ Like we saw last time, used Picasso to display the image in the card view widget

```
class DetailsActivity : BaseActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_details)  
        displayHomeAsUpEnabled(true)  
        val album: Album? = intent.extras?.getParcelable(key: "album")  
  
        txvAlbumName.text = "Album Name: {album?.name}"  
        txvAlbumArtist.text = "Artist Name: {album?.artist}"  
        txvAlbumPlayCount.text = "Play Count: {album?.playCount.toString()}"  
  
        Picasso.with(context: this@DetailsActivity).load(album?.image)  
            .error(R.drawable.ic_image_black_48dp)  
            .placeholder(R.drawable.ic_image_black_48dp)  
            .into(imvAlbumImage)  
    }  
}
```

RECYCLERITEMONCLICKLISTENER

- ▶ RecyclerView.OnClickListeners.kt implements RecyclerView.SimpleOnTouchListener
- ▶ GestureDetector.SimpleOnGestureListener
- ▶ Detects various gestures & events using the given **MotionEvent**
- ▶ onSingleTapUp override function
 - ▶ Notified when a tap occurs with the up **MotionEvent** that triggered it

```
class RecyclerView.OnClickListeners(
    context: Context,
    private val recyclerView: RecyclerView,
    private val listener: IRecyclerViewItem
) : RecyclerView.SimpleOnItemTouchListener() {

    private val gestureDetector = GestureDetectorCompat(context, object :
        GestureDetector.SimpleOnGestureListener() {
            override fun onSingleTapUp(e: MotionEvent): Boolean {
                val childView: View? = recyclerView.findChildViewUnder(e.x, e.y)
                if (childView != null) {
                    listener.onItemClick(childView, recyclerView.getChildAdapterPosition(childView))
                }
                return true
            }
        })

    override fun onInterceptTouchEvent(rv: RecyclerView, e: MotionEvent): Boolean {
        return gestureDetector.onTouchEvent(e)
    }
}
```

RECYCLERITEMONCLICKLISTENER - MAINACTIVITY

- ▶ Implemented in the onCreate
- ▶ addItemTouchListener - bind this to the RecyclerView widget
- ▶ Pass in the RecyclerViewItemClickListener object

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
    displayHomeAsUpEnabled( false)  
  
    lastFmRecyclerViewAdapter = LastFmRecyclerViewAdapter(ArrayList())  
    revAlbums.setLayoutManager( context: this@MainActivity)  
    revAlbums.addItemTouchListener(RecyclerViewItemClickListener( context: this@MainActivity, listener: this))  
    revAlbums.adapter = lastFmRecyclerViewAdapter  
  
    val url: String = createURI(getString(R.string.base_url), "artist.gettopalbums",  
        artist: "cher", "58384a2141a4b9737eacb9d0989b8a8c", "json")  
    rawDataAsyncTask = RawDataAsyncTask( listener: this)  
    rawDataAsyncTask.execute(url)  
}
```

IRECYCLERVIEWITEM

- One override function - onItemClick

```
interface IRecyclerViewItem {  
    fun onItemClick(view: View, position: Int)  
}
```

IRECYCLERVIEWITEM - MainActivity

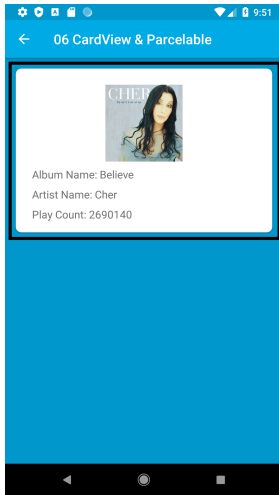
- Implemented in the MainActivity.kt. Of course, not limited to this class

```
override fun onItemClick(view: View, position: Int) {  
    val album: Album? = LastFmRecyclerViewAdapter.getAlbum(position)  
    if (album != null) {  
        val intent = Intent( packageContext: this, DetailsActivity::class.java)  
        intent.putExtra( name: "album", album)  
        startActivity(intent)  
    }  
}
```

CARD VIEW


- ▶ Containers that used in list views/recycler views to hold each item's information
- ▶ An easy & consistent way to show information in cards
- ▶ The card view has a default elevation above their containing view group
 - ▶ The system draws a box shadow below it
- ▶ Attributes to consider:
 - ▶ `cardElevation`
 - ▶ `cardCornerRadius`
 - ▶ `cardBackgroundColor`

CARD VIEW - EXAMPLE



PARCELABLE

- ▶ An interface used to serialize a class so its properties can be transferred from one activity to another
- ▶ Research task 1: What is the difference between a parcelable & serializable implementation?
- ▶ Research task 2: How to convert your Album data class into an Album parcelable class
- ▶ Research task 3: How to reduce the parcelable class boilerplate using @Parcelize



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

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: Today at 9am
- ▶ Deadline: Friday, 13 March at 10am