



# Lecture 08: Progress Dialog & Web View

## IN721: Design and Development of Applications for Mobile Devices

### Semester One, 2020

Kaiako: Grayson Orr

Te Kura Matatini ki Otago, Ōtepoti, Aotearoa

Friday, 13 March

# LECTURE 07: SEARCH VIEW & SHARED PREFERENCES

## TOPICS

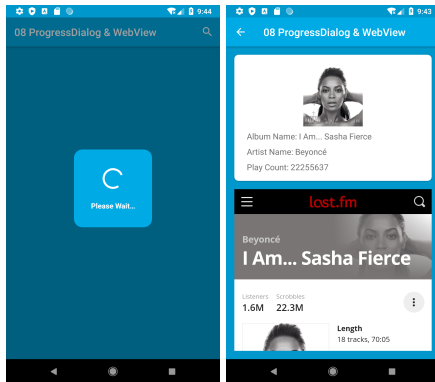
- ▶ File structure
- ▶ SearchActivity.kt
- ▶ Search view
  - ▶ menu\_search.xml
  - ▶ searchable.xml
  - ▶ AndroidManifest.xml
- ▶ Shared preferences

## LECTURE 08: PROGRESS DIALOG & WEB VIEW TOPICS

- ▶ Custom style
- ▶ Custom progress dialog/bar
- ▶ Web view

# APPLICATION SCREENSHOT

- ▶ Extension to the simple application using Last.fm API
- ▶ Last piece of functionality to add
  - ▶ 05 - async task & recycler view
  - ▶ 06 - card view & parcelable
  - ▶ 07 - search view & shared preferences



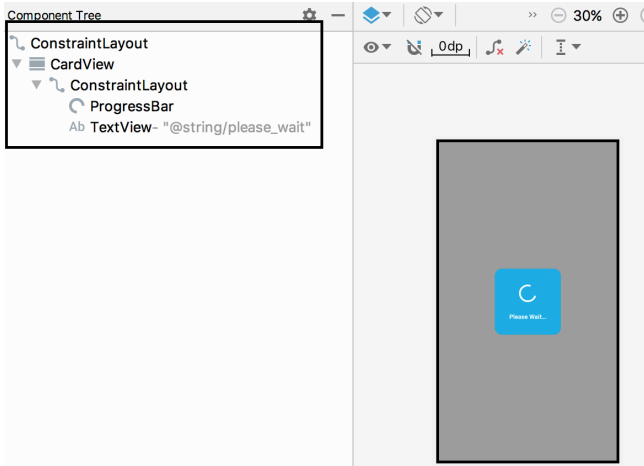
# CUSTOM STYLE

- ▶ styles.xml file
- ▶ Transparent window background

```
<style name="CustomProgressBarTheme">  
|   <item name="android:windowBackground">@android:color/transparent</item>  
</style>
```

# CUSTOM PROGRESS BAR: CARD VIEW

- Create a new layout file called progress\_bar.xml



# CUSTOM PROGRESS BAR: CLASS

- ▶ CustomProgressBar.kt
- ▶ Progress dialog deprecated in API level 26. Use progress bar instead
- ▶ A modal dialog preventing the user from interacting with the app
- ▶ Indicates the progress of an operation - downloading data from an API
  - ▶ Indeterminate - show when you do not know how long an operation will take
  - ▶ Determinate - you want to show that a specific quantity of progress has occurred

# CUSTOM PROGRESS BAR: CLASS

- ▶ Inflate the progress\_bar.xml layout file
- ▶ Create a **Dialog** object. Context & style XML are passed
- ▶ Two functions which show & hide the custom progress bar

```
class CustomProgressBar(context: Context) {  
    private val inflater: LayoutInflater = context.getSystemService(Context.LAYOUT_INFLATER_SERVICE) as LayoutInflater  
    private val view: View = inflater.inflate(R.layout.progress_bar, root: null)  
    private val dialog = Dialog(context, R.style.CustomProgressBarTheme)  
  
    fun show() {  
        dialog setContentView(view)  
        dialog.show()  
    }  
  
    fun dismiss() {  
        dialog.dismiss()  
    }  
}
```



# CUSTOM PROGRESS BAR: ASYNC TASK

- ▶ Pass context as a class argument
- ▶ Context - reference to the class that will use this async task (**DetailsActivity.kt**)
- ▶ Show the progress bar when downloading data
- ▶ Hide/dismiss the progress bar when finished downloading data

```
class RawDataAsyncTask(private val listener: IDownloadComplete, context: Context)
: AsyncTask<String, Void, String>() {
    private var downloadStatus = DownloadStatus.NONE
    private var progressBar = CustomProgressBar(context)

    override fun onPreExecute() {
        progressBar.show()
    }

    override fun onPostExecute(result: String) {
        progressBar.dismiss()
        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()
    }
}
```

# CUSTOM PROGRESS BAR: DETAILSACTIVITY

- ▶ Show the progress bar when the **DetailsActivity** is created
- ▶ Hide/dismiss the progress bar when the web view has loaded

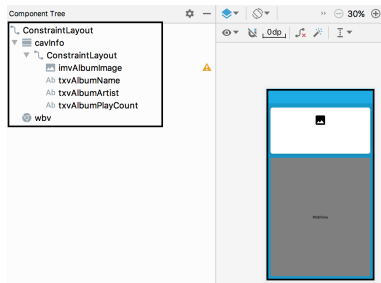
```
class DetailsActivity : BaseActivity() {  
    private lateinit var progressBar: CustomProgressBar  
    private lateinit var album: Album  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_details)  
        displayHomeAsUpEnabled(true)  
  
        progressBar = CustomProgressBar(context = this@DetailsActivity)  
        album = intent.extras.getParcelable<Album>("album")  
        webView.settings.javaScriptEnabled = true  
        webView.loadUrl(album.url)  
        webView.webViewClient = webViewClient  
  
        progressBar.show()  
    }  
  
    private val webViewClient: WebViewClient = object: WebViewClient() {  
        override fun onPageFinished(view: WebView?, url: String?) {  
            super.onPageFinished(view, url)  
            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(invAlbumImage)  
  
            progressBar.dismiss()  
        }  
    }  
}
```

# WEB VIEW

- ▶ Recommended - standard web browser instead of web view
- ▶ Invoke a browser with an intent
  - ▶ ACTION\_VIEW
- ▶ Display web content as part of your activity layout

# WEB VIEW

- ▶ Add a web view widget to content\_details.xml
- ▶ The widget place positioned below the card view



# WEB VIEW

- ▶ Write web view code in DetailsActivity.kt
- ▶ WebViewClient object
- ▶ onPageFinished override function
  - ▶ Two arguments - view & url
  - ▶ Notify the host application that a page has finished loading

```
class DetailsActivity : BaseActivity() {  
    private lateinit var progressBar: CustomProgressBar  
    private lateinit var album: Album  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_details)  
        displayHomeAsUpEnabled(true)  
  
        progressBar = CustomProgressBar(context = this@DetailsActivity)  
  
        album = Intent.extras.getParcelable(key = "album")  
  
        webView.settings.javaScriptEnabled = true  
        webView.loadUrl(album.url)  
        webView.webViewClient = webViewClient  
  
        progressBar.show()  
    }  
  
    private val webViewClient: WebViewClient = object: WebViewClient() {  
        override fun onPageFinished(view: WebView?, url: String?) {  
            super.onPageFinished(view, url)  
            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(imgAlbumImage)  
  
            progressBar.dismiss()  
        }  
    }  
}
```

# 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
- ▶ Deadline: Today at 10am
- ▶ Feedback & results will be available tonight via Microsoft Teams

# NEXT WEEK

- ▶ Alert dialog builder & fragments
- ▶ Pair programming practicals