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

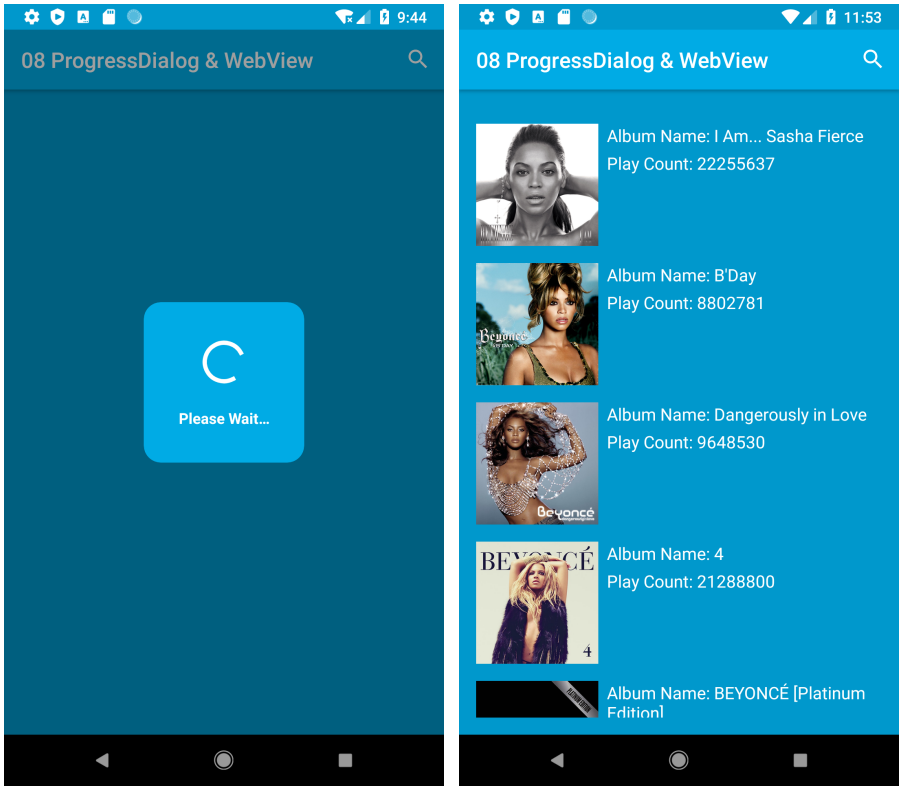
- File structure
- SearchActivity.kt
- Search view
  - menu_search.xml
  - searchable.xml
  - AndroidManifest.xml
- Shared preferences

Lecture 08: Progress Dialog & Web View Topics

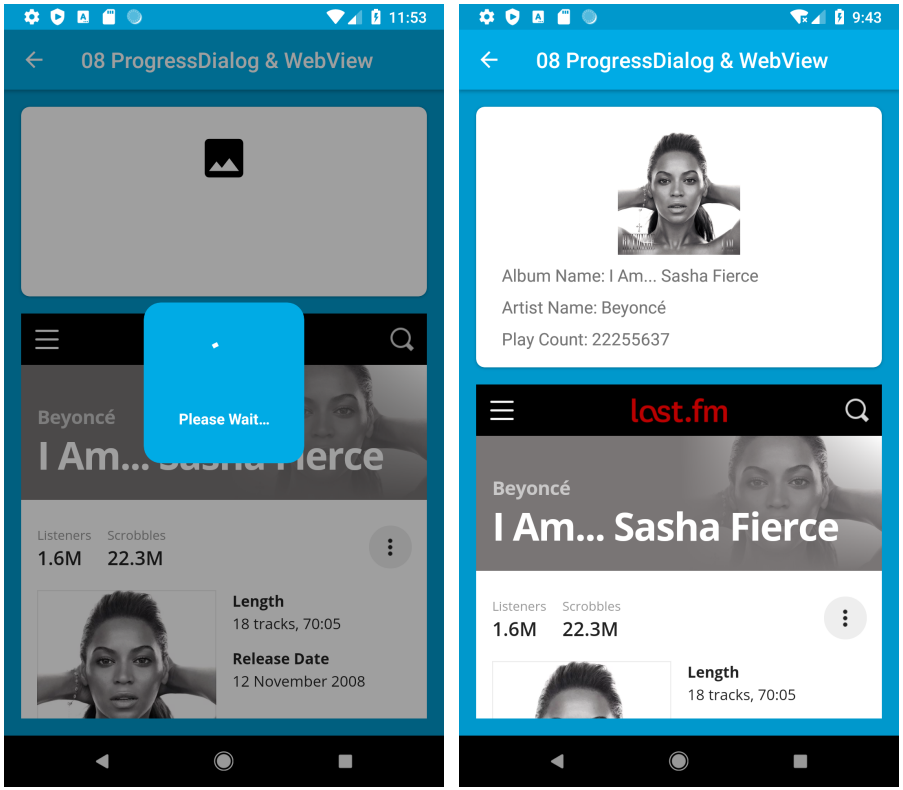
- Custom style
- Custom progress dialog/bar
- Web view

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



Application Screenshot



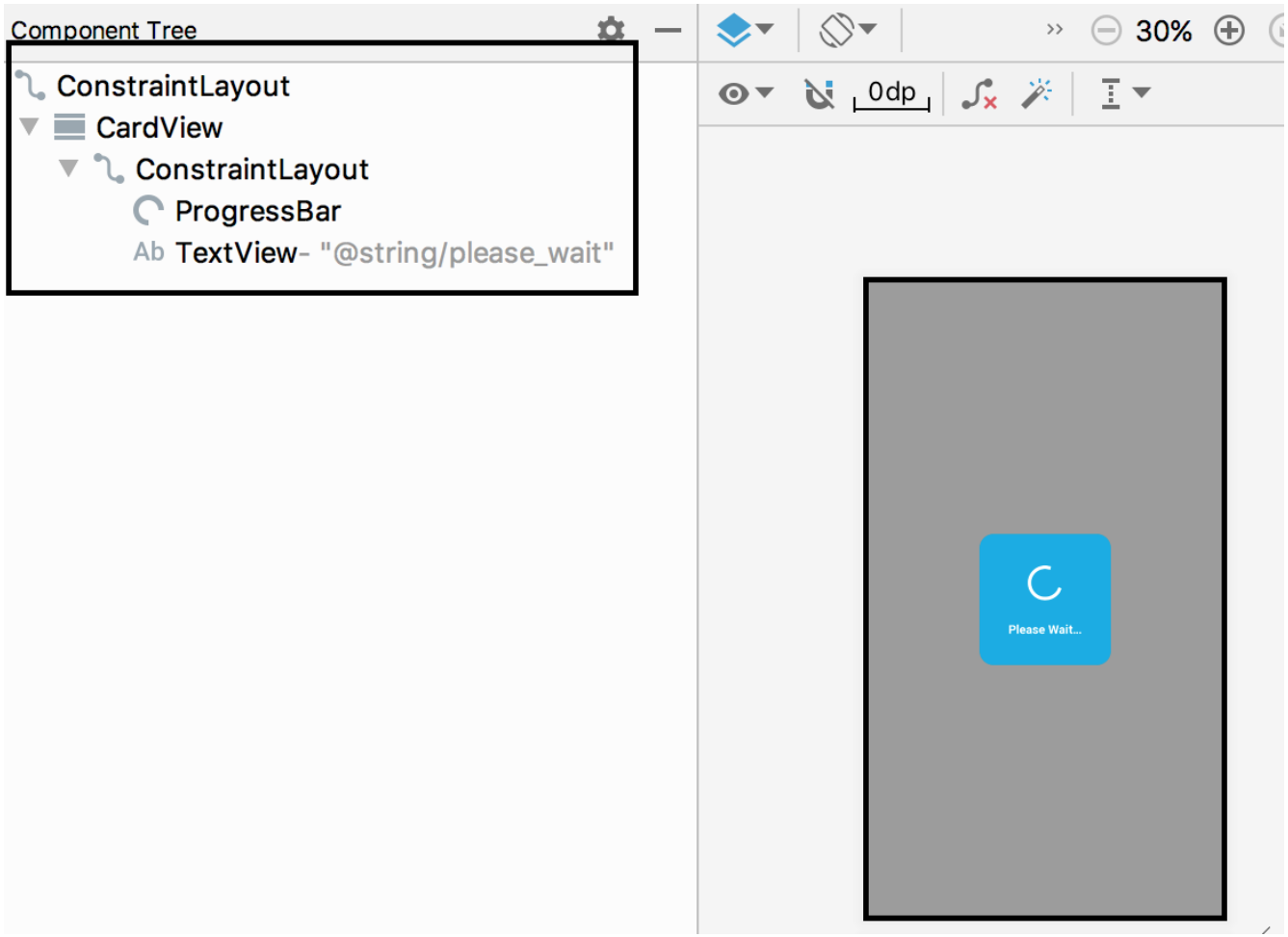
Custom Style

- 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: IDataDownloadComplete, 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

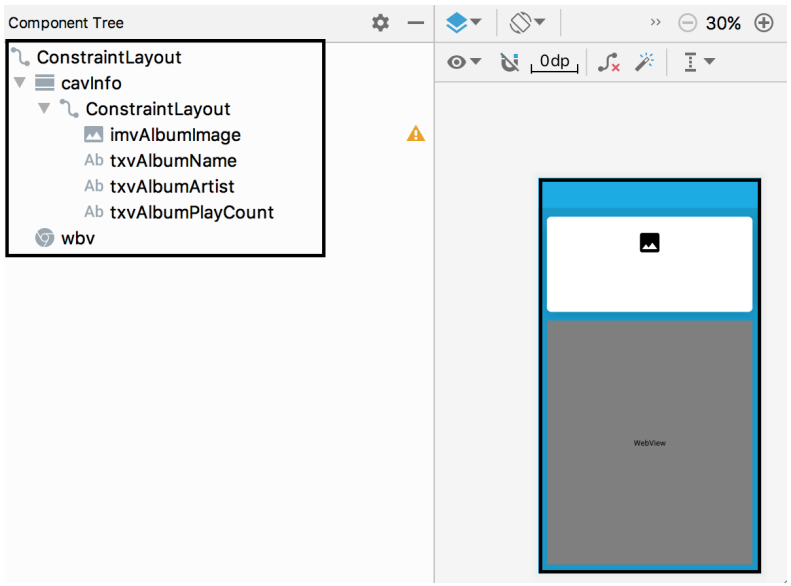
```
class DetailsActivity : BaseActivity() {
    private lateinit var progressBar: CustomProgressBar
    private lateinit var album: Album

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_details)
        displayToolbar( isHomeEnabled: true)

        progressBar = CustomProgressBar( context: this@DetailsActivity)

        album = intent.extras.getParcelable( key: "album")

        wbv.settings.javaScriptEnabled = true
        wbv.loadUrl(album.url)
        wbv.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(imvAlbumImage)

            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)
        displayToolbar( isHomeEnabled: true)

        progressBar = CustomProgressBar( context: this@DetailsActivity)

        album = intent.extras.getParcelable( key: "album")

        wbv.settings.javaScriptEnabled = true
        wbv.loadUrl(album.url)
        wbv.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(imvAlbumImage)

            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 questions 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