



Lecture 07: Search View & Shared Preferences

IN721: Design and Development of Applications for Mobile Devices
Semester One, 2020

Kaiako: Grayson Orr

Te Kura Matatini ki Otago, Ōtepoti, Aotearoa

Wednesday, 11 March

LECTURE 06: CARD VIEW & PARCELABLE TOPICS

- ▶ File structure
- ▶ Card view
- ▶ Parcelable

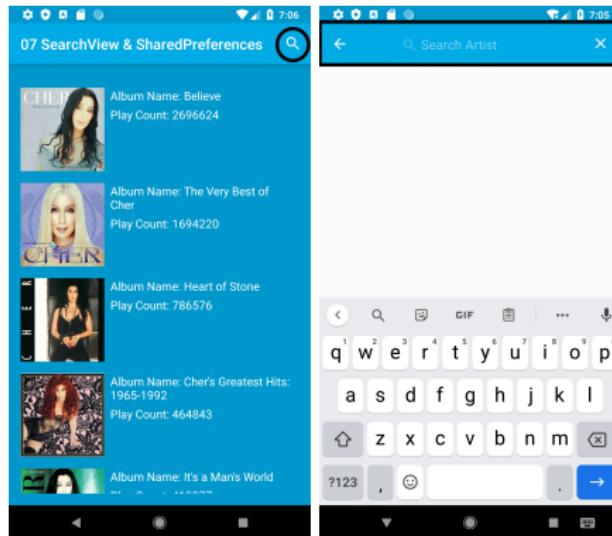
LECTURE 07: SEARCH VIEW & SHARED PREFERENCES

TOPICS

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

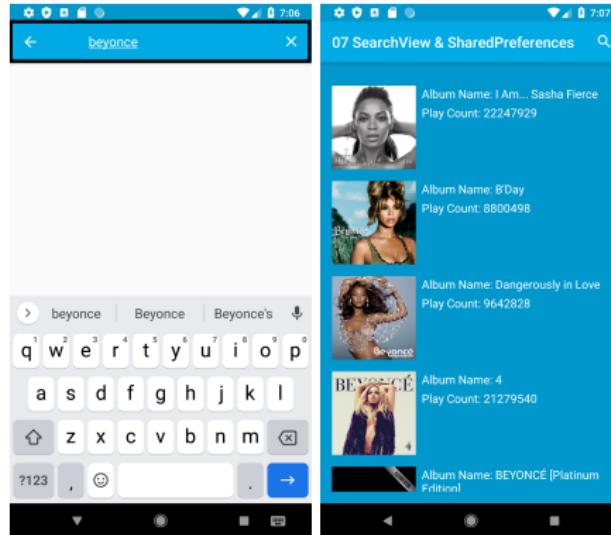
APPLICATION SCREENSHOT

- Recycler view of Cher albums
- Search icon in the top right-hand corner of the application
- Click on the search icon - start new activity called **SearchActivity**



APPLICATION SCREENSHOT

- ▶ Type **beyonce** into the search view toolbar
- ▶ Activity is destroyed & starts new activity - MainActivity
- ▶ Recycler view of Beyonce albums



FILE STRUCTURE

- ▶ Adding one file to our current structure:
 - ▶ SearchActivity.kt - activities directory
 - ▶ Create a basic activity
 - ▶ Delete floating action button from activity XML file
 - ▶ Delete content XML file

SEARCHACTIVITY.KT

- SearchActivity implements BaseActivity

```
class SearchActivity : BaseActivity() {  
    private lateinit var searchView: SearchView  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_search)  
        displayToolbar{ isHomeEnabled: true}  
    }  
  
    override fun onCreateOptionsMenu(menu: Menu): Boolean {  
        menuInflater.inflate(R.menu.menu_search, menu)  
  
        val searchManager: SearchManager = getSystemService(Context.SEARCH_SERVICE) as SearchManager  
        val searchableInfo: SearchableInfo = searchManager.getSearchableInfo(componentName)  
  
        searchView = menu.findItem(R.id.action_search).actionView as SearchView  
        searchView.setSearchableInfo(searchableInfo)  
        searchView.isIconified = false  
  
        searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
            override fun onQueryTextSubmit(query: String): Boolean {  
                val sharedPref: SharedPreferences = PreferenceManager.getDefaultSharedPreferences(applicationContext)  
                sharedPref.edit().putString("album_query", query).apply()  
                searchView.clearFocus()  
                finish()  
                return true  
            }  
  
            override fun onQueryTextChange(newText: String?): Boolean {  
                return false  
            }  
        })  
  
        searchView.setOnCloseListener {  
            finish()  
            false /*setOnCloseListener*/  
        }  
    }  
  
    return true  
}
```

SEARCHACTIVITY.KT

- ▶ Inflate the menu_search.xml layout
- ▶ SearchManager - provides access to the system search services
- ▶ SearchableInfo - only applications that search other applications should need to use this class

```
class SearchActivity : BaseActivity() {  
    private lateinit var searchView: SearchView  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_search)  
        displayToolbar{ isHomeEnabled: true}  
    }  
  
    override fun onCreateOptionsMenu(menu: Menu): Boolean {  
        menuInflater.inflate(R.menu.menu_search, menu)  
  
        val searchManager = getSystemService(Context.SEARCH_SERVICE) as SearchManager  
        val searchableInfo: SearchableInfo = searchManager.getSearchableInfo(componentName)  
  
        searchView = menu.findItem(R.id.action_search).actionView as SearchView  
        searchView.setSearchableInfo(searchableInfo)  
        searchView.isIconified = false  
  
        searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
            override fun onQueryTextSubmit(query: String?): Boolean {  
                val sharedPref: SharedPreferences = PreferenceManager.getDefaultSharedPreferences(applicationContext)  
                sharedPref.edit().putString("album_query", query).apply()  
                searchView.clearFocus()  
                finish()  
                return true  
            }  
  
            override fun onQueryTextChange(newText: String?): Boolean {  
                return false  
            }  
        })  
  
        searchView.setOnCloseListener {  
            finish()  
            false //setOnCloseListener  
        }  
    }  
    return true  
}
```

SEARCHACTIVITY.KT

- ▶ Declare search view widget - late initialisation
- ▶ Find search icon menu item & bind to search view
- ▶ setSearchableInfo - display labels, hints, suggestions
- ▶ isIconified - return the current iconified state of the search view
 - ▶ Focuses on search view when activity is launched
 - ▶ If set to true, you need to click on the search icon

```
class SearchActivity : BaseActivity() {  
    private lateinit var searchView: SearchView  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_search)  
        setSupportActionBar(toolbar, isIconified: true)  
    }  
  
    override fun onCreateOptionsMenu(menu: Menu): Boolean {  
        menuInflater.inflate(R.menu.menu_search, menu)  
  
        val searchManager: SearchManager = getSystemService(Context.SEARCH_SERVICE) as SearchManager  
        val searchableInfo: SearchableInfo? = searchManager.getSearchableInfo(componentName)  
  
        searchView = menu.findItem(R.id.action_search).actionView as SearchView  
        searchView.setSearchableInfo(searchableInfo)  
        searchView.isIconified = false  
  
        searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
            override fun onQueryTextSubmit(query: String?): Boolean {  
                val sharedPref: SharedPreferences = PreferenceManager.getDefaultSharedPreferences(applicationContext)  
                val editor: SharedPreferences.Editor = sharedPref.edit().putString("alblue_query", query).apply()  
                searchView.clearFocus()  
                finish()  
                return true  
            }  
  
            override fun onQueryTextChange(newText: String?): Boolean {  
                return false  
            }  
        })  
  
        searchView.setOnCloseListener {  
            finish()  
            false //setOnCloseListener  
        }  
    }  
  
    return true  
}
```

SEARCHACTIVITY.KT

- ▶ `setOnQueryTextListener` - sets a listener for user actions within the search view
- ▶ Two override functions
 - ▶ `onQueryTextSubmit` - called when the user submits the query
 - ▶ `onQueryTextChange` - called when the query text is changed by the user

```
class SearchActivity : BaseActivity() {  
    private lateinit var searchView: SearchView  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_search)  
        displayToolbar(isHomeEnabled: true)  
    }  
  
    override fun onCreateOptionsMenu(menu: Menu): Boolean {  
        menuInflater.inflate(R.menu.menu_search, menu)  
  
        val searchManager: SearchManager = getSystemService(Context.SEARCH_SERVICE) as SearchManager  
        val searchableInfo: SearchableInfo? = searchManager.getSearchableInfo(componentName)  
  
        searchView = menu.findItem(R.id.action_search).actionView as SearchView  
        searchView.setSearchableInfo(searchableInfo)  
        searchView.isIconified = false  
  
        searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
            override fun onQueryTextSubmit(query: String?): Boolean {  
                val sharedPref: SharedPreferences = PreferenceManager.getDefaultSharedPreferences(applicationContext)  
                sharedPref.edit().putString("album_query", query).apply()  
                searchView.clearFocus()  
                finish()  
                return true  
            }  
  
            override fun onQueryTextChange(newText: String?): Boolean {  
                return false  
            }  
        })  
    }  
  
    searchView.setOnCloseListener {  
        finish()  
        false "setOnCloseListener"  
    }  
  
    return true  
}
```

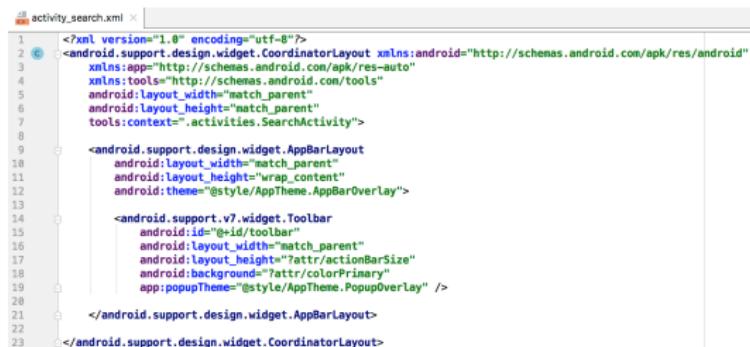
SEARCHACTIVITY.KT

- `setOnCloseListener` - sets a listener to inform when the user closes the `SearchView`

```
class SearchActivity : BaseActivity() {  
    private lateinit var searchView: SearchView  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_search)  
        displayToolbar(allowHomeEnabled: true)  
    }  
  
    override fun onCreateOptionsMenu(menu: Menu): Boolean {  
        menuInflater.inflate(R.menu.menu_search, menu)  
  
        val searchManager: SearchManager = getSystemService(Context.SEARCH_SERVICE) as SearchManager  
        val searchableInfo: SearchableInfo? = searchManager.getSearchableInfo(componentName)  
  
        searchView = menu.findItem(R.id.action_search).actionView as SearchView  
        searchView.setSearchableInfo(searchableInfo)  
        searchView.isIconified = false  
  
        searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
            override fun onQueryTextSubmit(query: String?): Boolean {  
                val sharedPref: SharedPreferences = PreferenceManager.getDefaultSharedPreferences(applicationContext)  
                sharedPref.edit().putString("album_query", query).apply()  
                searchView.clearFocus()  
                finish()  
                return true  
            }  
  
            override fun onQueryTextChange(newText: String?): Boolean {  
                return false  
            }  
        })  
  
        searchView.setOnCloseListener {  
            finish()  
            false //setOnCloseListener  
        }  
    }  
    return true  
}
```

ACTIVITY_SEARCH.XML

- Boilerplate activity XML file - you only need the toolbar widget



The screenshot shows the code editor of an Android Studio project. The file being viewed is 'activity_search.xml'. The code is an XML configuration for an activity. It includes declarations for CoordinatorLayout, AppBarLayout, Toolbar, and CoordinatorLayout again. The Toolbar is identified by the ID '@+id/toolbar'.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activities.SearchActivity">
    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">
        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />
    </android.support.design.widget.AppBarLayout>
</android.support.design.widget.CoordinatorLayout>
```

SEARCH VIEW: MENU LAYOUTS

- ▶ menu_main.xml & menu_search.xml
- ▶ Menu resources directory
- ▶ app:actionViewClass attribute
 - ▶ Class of a widget that implements the action - SearchView widget



```
menu_main.xml
1 <menu xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:app="http://schemas.android.com/apk/res-auto"
3   xmlns:tools="http://schemas.android.com/tools"
4   tools:context=".activities.MainActivity">
5
6   <item
7     android:id="@+id/action_main"
8     android:icon="@drawable/ic_search_white_24dp"
9     android:title="Main"
10    app:showAsAction="ifRoom" />
11
menu_search.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto">
4
5   <item
6     android:id="@+id/action_search"
7     android:icon="@drawable/ic_search_white_24dp"
8     android:title="Search"
9     app:actionViewClass="android.widget.SearchView"
10    app:showAsAction="ifRoom" />
11
```

SEARCH VIEW: MENU_MAIN.XML

- ▶ Inflate the menu_main.xml layout in **MainActivity.kt**
- ▶ Start a new activity when the search icon is clicked

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    menuInflater.inflate(R.menu.menu_main, menu)
    return true
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    return when (item.itemId) {
        R.id.action_main -> {
            startActivity(Intent( packageContext: this@MainActivity, SearchActivity::class.java))
            true
        }
        else -> super.onOptionsItemSelected(item)
    }
}
```

SEARCH VIEW: SEARCHABLE.XML

- ▶ Create a new resource directory called xml
- ▶ Create a xml file called searchable.xml



The screenshot shows an XML editor window with the title bar "searchable.xml". The code area contains the following XML:

```
<?xml version="1.0" encoding="utf-8"?>
<searchable xmlns:android="http://schemas.android.com/apk/res/android"
    android:hint="Search Artist"
    android:label="Search Artist" />
```

SEARCH VIEW: ANDROIDMANIFEST.XML

- ▶ intent-filter
 - ▶ android.intent.action.SEARCH
- ▶ meta-data
 - ▶ android:resource="@xml/searchable"

```
<activity
    android:name=".activities.SearchActivity"
    android:parentActivityName=".activities.MainActivity"
    android:theme="@style/AppTheme.NoActionBar">
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="graysono.com.cp08searchviewsharedpreferences.activities.MainActivity"/>
    <intent-filter>
        <action android:name="android.intent.action.SEARCH"/>
    </intent-filter>
    <meta-data android:name="android.app.searchable"
        android:resource="@xml/searchable"/>
</activity>
```

SHARED PREFERENCES

- ▶ If you have a relatively small collection of key-values that you'd like to save
- ▶ SharedPreferences object points to a file containing key/value pairs
- ▶ Provides methods to read and write
- ▶ SharedPreferences file is managed by the framework & can be private or shared
 - ▶ For simplicity, we will use shared

SHARED PREFERENCES

- ▶ album_query key/value data passed from **SearchActivity.kt** to **MainActivity.kt**
- ▶ PreferenceManager.getDefaultSharedPreferences - points to the default file that is used by the preference framework
 - ▶ this@SearchActivity or applicationContext

```
searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {  
    override fun onQueryTextSubmit(query: String?): Boolean {  
        val sharedPref: SharedPreferences = PreferenceManager.getDefaultSharedPreferences(applicationContext)  
        sharedPref.edit().putString("album_query", query).apply()  
        searchView.clearFocus()  
        finish()  
        return true  
    }  
  
    override fun onQueryTextChange(newText: String?): Boolean {  
        return false  
    }  
})
```

SHARED PREFERENCES: OnResume

- ▶ In **MainActivity.kt**
- ▶ Check for value in album_query key
- ▶ If there is data, download the artist's album data from Last.fm via async task
- ▶ Display the data in the recycler view

```
override fun onResume() {
    super.onResume()
    val sharedPref: SharedPreferences = PreferenceManager.getDefaultSharedPreferences(applicationContext)
    val queryResult: String? = sharedPref.getString("album_query", "")

    if (queryResult!!.isNotEmpty()) {
        val url: String = createURI(
            "http://ws.audioscrobbler.com/2.0/", "artist.gettopalbums",
            queryResult, "58384a2141a4b9737eacb9d0989b8a8c", "json"
        )
        rawDataAsyncTask = RawDataAsyncTask( listener: this)
        rawDataAsyncTask.execute(url)
    }
}
```

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: Friday, 13 March at 10am