twitch.tv/dancojocar

youtube.com/dancojocar

Lecture #3
Navigation and Rest Resources

Mobile Applications
Fall 2024

REST using Retrofit

Retrofit

A type-safe **HTTP client** for Android and Java

Introduction

Retrofit turns your HTTP API into a Java interface.

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

The `Retrofit` class generates an implementation of the `GitHubService` interface.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com/")
    .build();

GitHubService service = retrofit.create(GitHubService.class);
```

Each `Call` from the created `GitHubService` can make a synchronous or asynchronous HTTP request to the remote webserver.

```
Call<List<Repo>> repos = service.listRepos("octocat");
```

Use annotations to describe the HTTP request:

- URL parameter replacement and query parameter support
- Object conversion to request body (e.g., JSON, protocol buffers)
- Multipart request body and file upload

https://square.github.io/retrofit/

REST using Retrofit

```
implementation "com.squareup.retrofit2:retrofit:version"
implementation "com.squareup.retrofit2:adapter-rxjava2:version"
implementation "com.squareup.retrofit2:converter-gson:version"

implementation “io.reactivex.rxjava2:rxandroid:version"
```

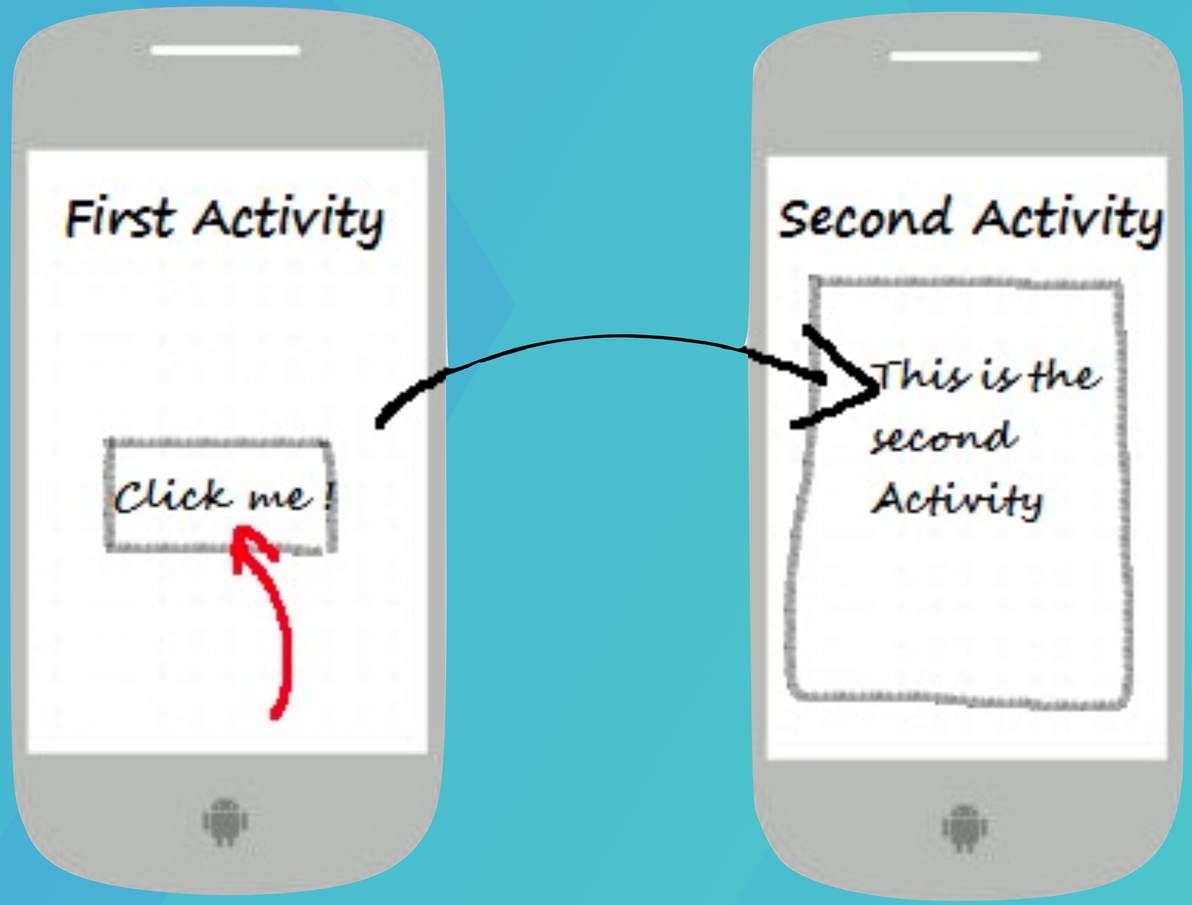
https://square.github.io/retrofit/

REST using Retrofit


```
interface MovieService {

  @GET("movies")
  val movies: Observable<List<Movie>>

  @GET("genres")
  val genres: Observable<List<String>>

  @GET("moviesByGenre/{genre}")
  fun moviesByGenre(@Path("genre") genre: String)
      : Observable<List<Movie>>

  @GET("details/{id}")
  fun details(@Path("id") id: Int): Observable<Movie>

  @POST("updateDescription")
  fun updateDescription(@Body movie: Movie): Observable<Movie>

  @POST("updateRating")
  fun updateRating(@Body movie: Movie): Observable<Movie>

  @POST("update")
  fun update(@Body movie: Movie): Observable<Movie>

  @DELETE("delete/{id}")
  fun delete(@Path("id") id: Int): Observable<ResponseBody>

  @POST("add")
  fun add(@Body movie: Movie): Observable<Movie>

  companion object {
    const val SERVICE_ENDPOINT = “http://SERVER_IP:2022”
  }
}
```

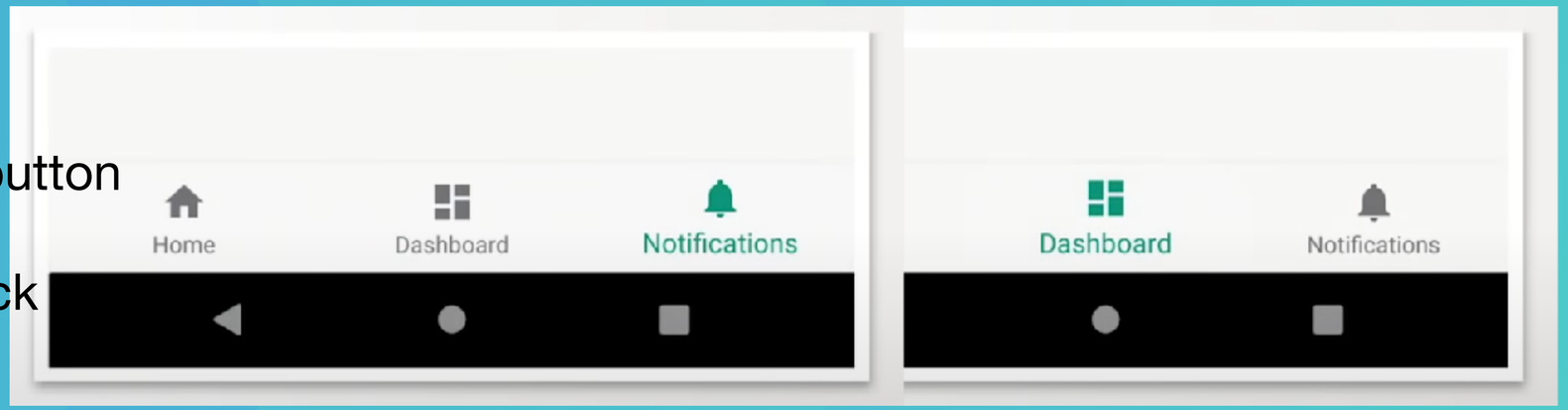
https://square.github.io/retrofit/

Navigation


First Activity
Click me !
Second Activity
This is the second Activity

Navigation

- Correctly
- Highlight the correct button
- Handles the back-stack



Key concepts

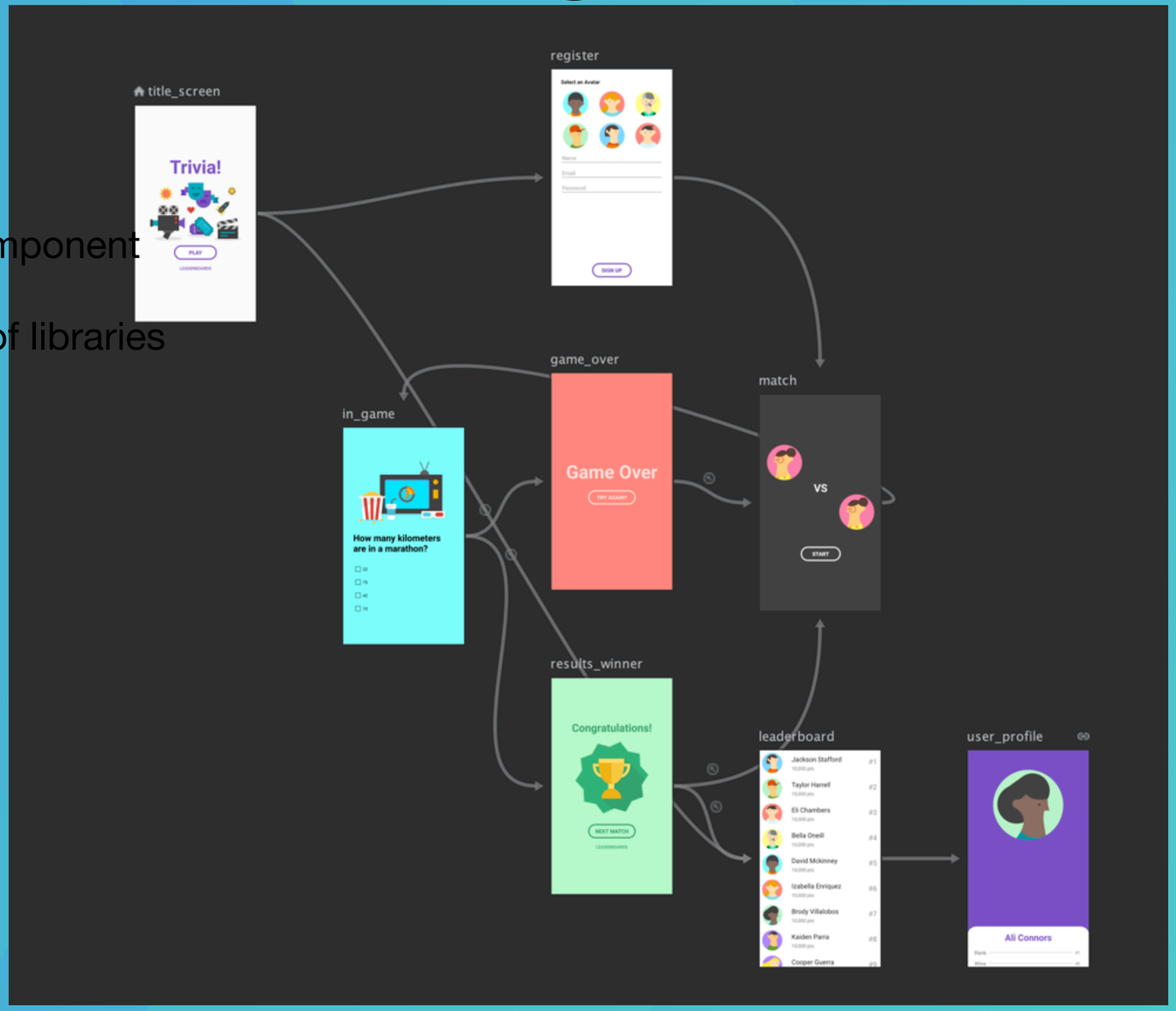
Concept	Purpose	Type
Host	A UI element that contains the current navigation destination. That is, when a user navigates through an app, the app essentially swaps destinations in and out of the navigation host.	• Compose: `NavHost` • Fragments: `NavHostFragment`
Graph	A data structure that defines all the navigation destinations within the app and how they connect together.	`NavGraph`
Controller	The central coordinator for managing navigation between destinations. The controller offers methods for navigating between destinations, handling deep links, managing the back stack, and more.	`NavController`
Destination	A node in the navigation graph. When the user navigates to this node, the host displays its content.	`NavDestination` Typically created when constructing the navigation graph.
Route	Uniquely identifies a destination and any data required by it. You can navigate using routes. Routes take you to destinations.	Any serializable data type.

https://developer.android.com/guide/navigation

Navigation

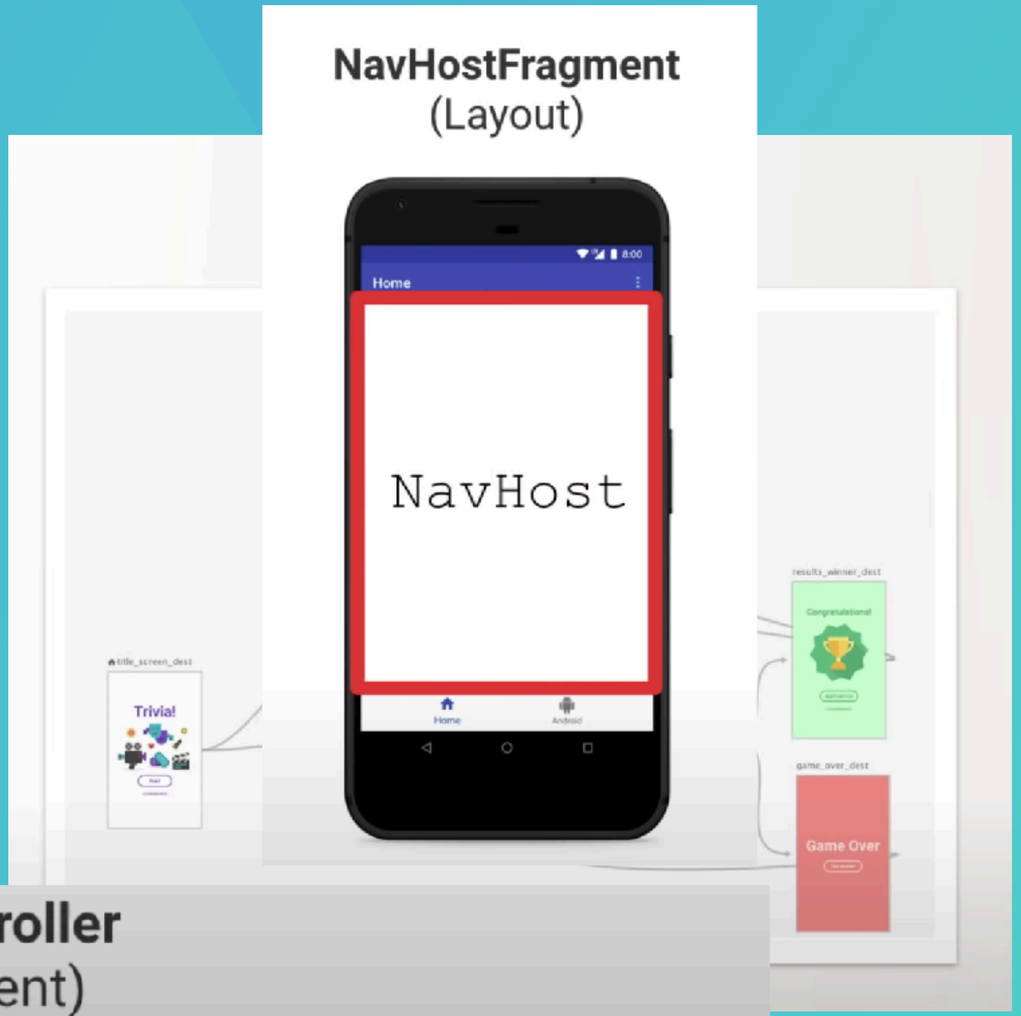
Navigation Component

- A collection of libraries
- A plugin
- Tooling



Navigation

- Navigation Graph
- NavHostFragment
- NavController



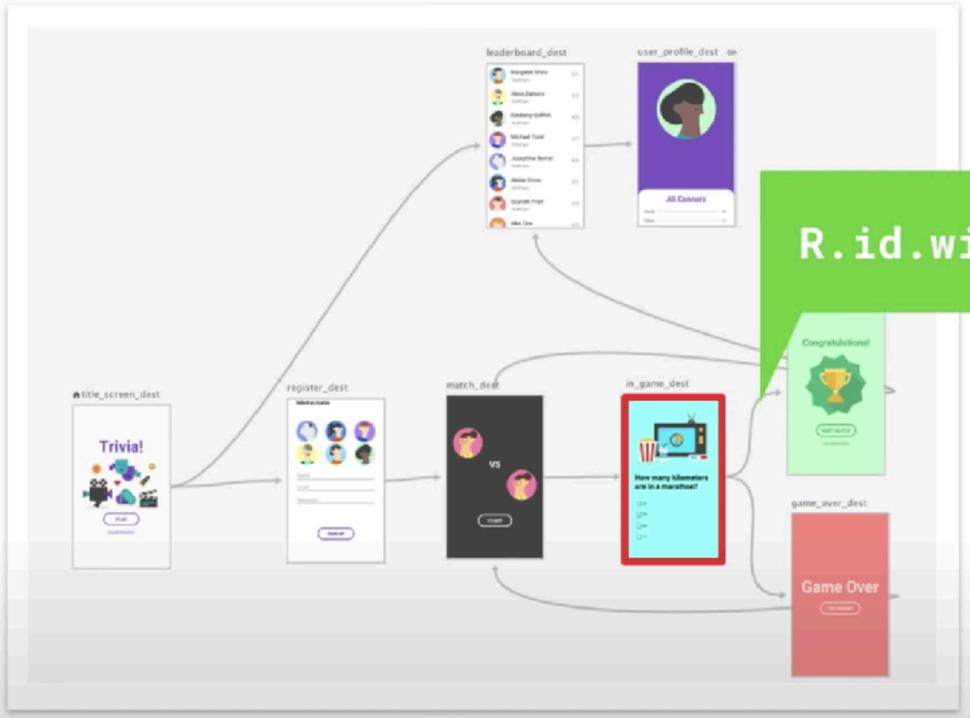
NavController
(Fragment)

```
findNavController().navigate(<Destination or Action id>)
```

Navigation

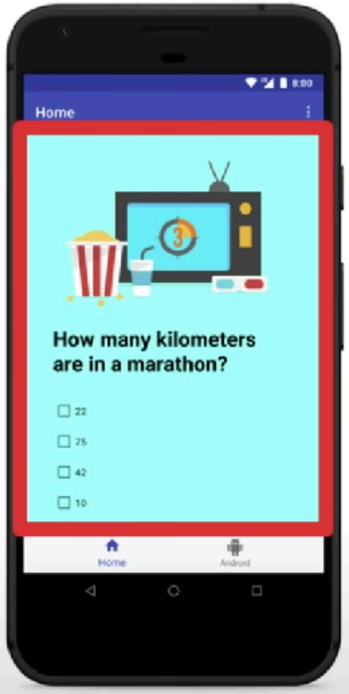
DEMO

Navigation Graph
(New Resource)



R.id.win_action

NavHostFragment
(Layout)



NavController
(Fragment)

```
findNavController().navigate(R.id.win_action)
```

https://developer.android.com/guide/navigation

Navigation with Compose

```
dependencies {
    val nav_version = "2.8.2"

    implementation("androidx.navigation:navigation-compose:$nav_version")
}
```

https://developer.android.com/develop/ui/compose/navigation

Navigate to a composable

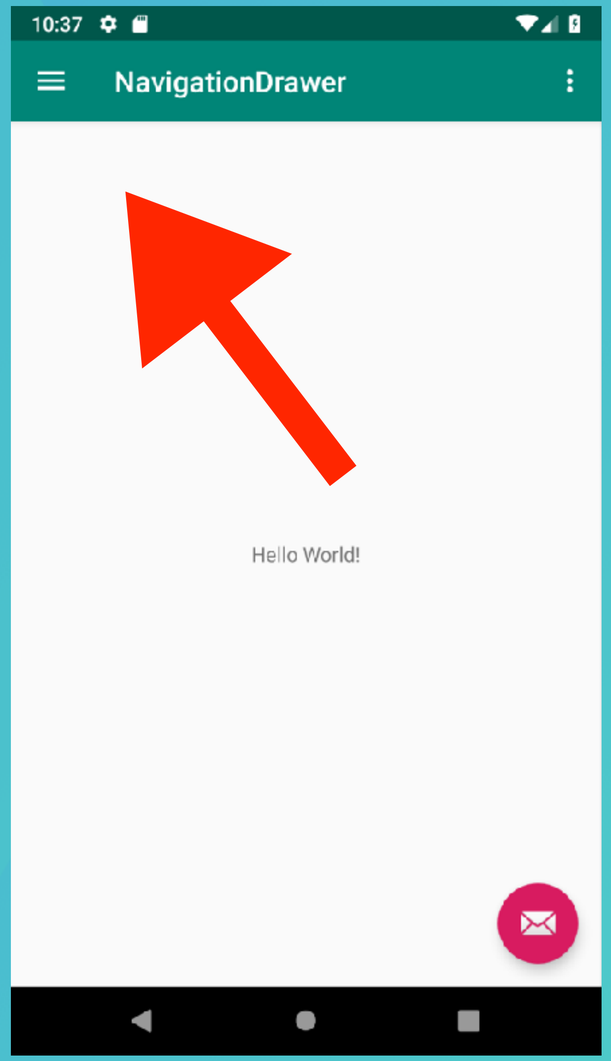
```
@Serializable
object FriendsList

navController.navigate(route = FriendsList)
```

```
@Serializable
object Profile
@Serializable
object FriendsList
@Composable
fun MyAppNavHost(
    modifier: Modifier = Modifier,
    navController: NavHostController = rememberNavController(),
) {
    NavHost(
        modifier = modifier,
        navController = navController,
        startDestination = Profile
    ) {
        composable<Profile> {
            ProfileScreen(
                onNavigateToFriends = { navController.navigate(route = FriendsList) },
                /*...*/
            )
        }
        composable<FriendsList> { FriendsListScreen(/*...*/) }
    }
}
@Composable
fun ProfileScreen(
    onNavigateToFriends: () -> Unit,
    /*...*/
) {
```

```
fun MyAppNavHost(
    modifier: Modifier = Modifier,
    navController: NavHostController = rememberNavController(),
) {
    NavHost(
        modifier = modifier,
        navController = navController,
        startDestination = Profile
    ) {
        composable<Profile> {
            ProfileScreen(
                onNavigateToFriends = { navController.navigate(route = FriendsList) },
                /*...*/
            )
        }
        composable<FriendsList> { FriendsListScreen(/*...*/) }
    }
}
@Composable
fun ProfileScreen(
    onNavigateToFriends: () -> Unit,
    /*...*/
) {
    /*...*/
    Button(onClick = onNavigateToFriends) {
        Text(text = "See friends list")
    }
}
```

Navigation Drawer

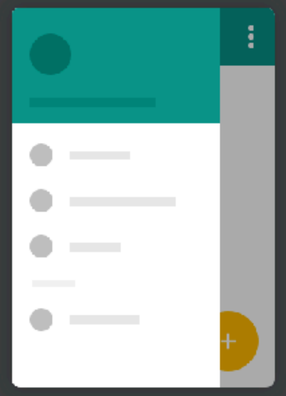
- App main navigation menu.
- Hidden when not in use.
- Appears:
  - with a left swipe from the screen edge
  - when the user touches the drawer icon in the app bar



https://developer.android.com/training/implementing-navigation/nav-drawer

Create New Project

Configure Your Project



Navigation Drawer Activity

Creates a new Activity with a Navigation Drawer

Name

My Application

Package name

ro.cojocar.myapplication

Save location

/Users/dan/work/school/ma/lectures/3/MyApplication

Language

Kotlin

Minimum SDK API 29: Android 10.0 (Q)

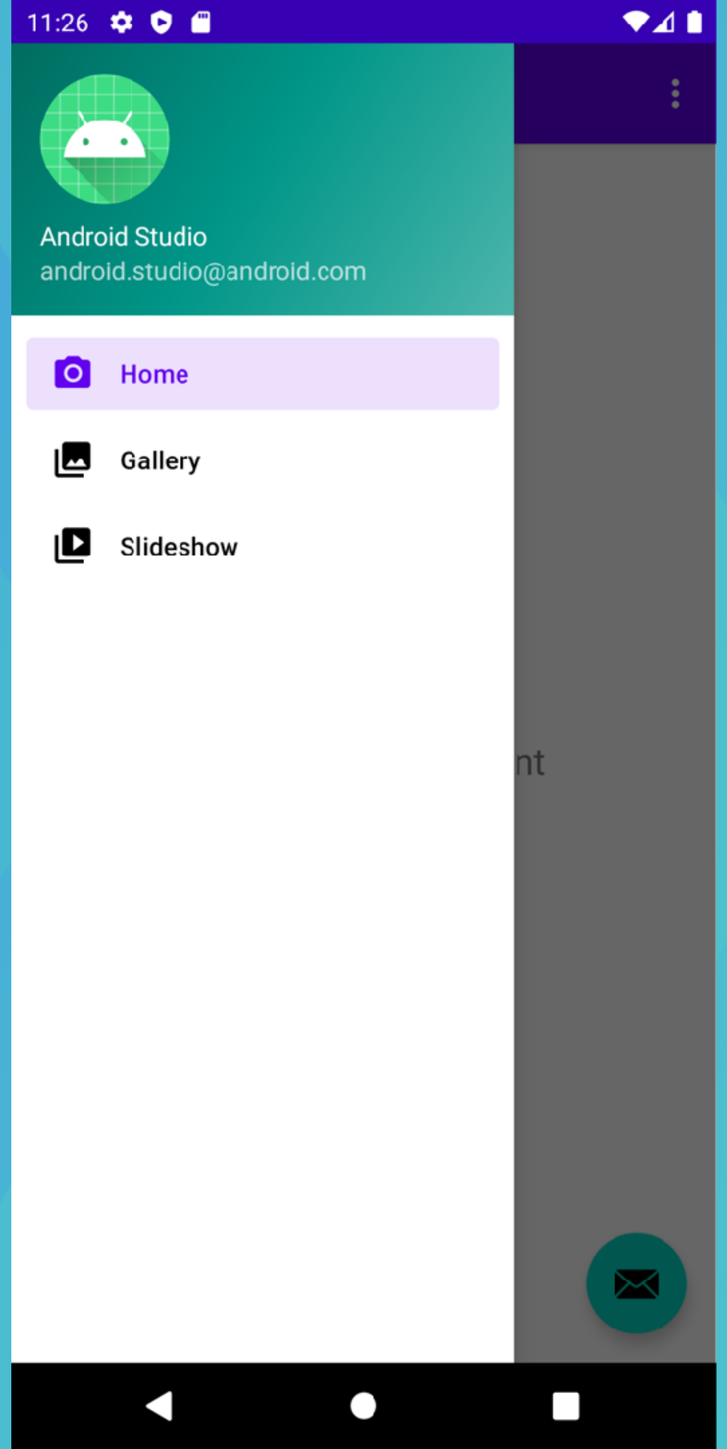
Your app will run on approximately 8.2% of devices.
Help me choose

Use legacy android.support libraries

Cancel Previous Next Finish

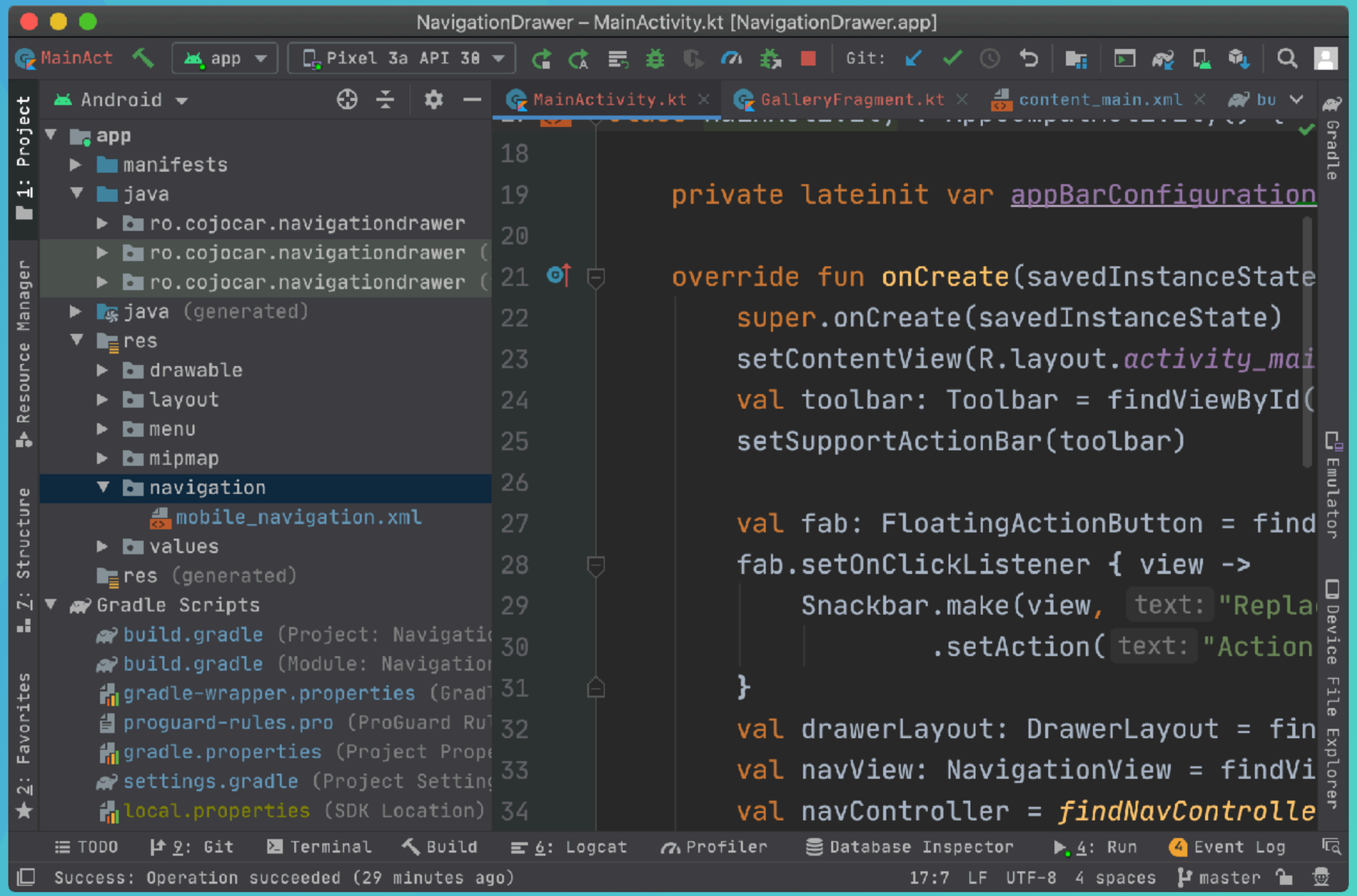
Gallery...

Android TV Activity (Requires
Android Things Empty Activity
Android Things Peripheral Acti
Basic Activity
Blank Wear Activity (Requires
Bottom Navigation Activity
Empty Activity
Fragment + ViewModel
Fullscreen Activity
Login Activity
Master/Detail Flow
Navigation Drawer Activity
Scrolling Activity
Settings Activity
Tabbed Activity


11:26
Android Studio
android.studio@android.com
Home
Gallery
Slideshow

Generated Artifacts

- Sources
- Layouts
- Menus
- Navigation



Dependencies

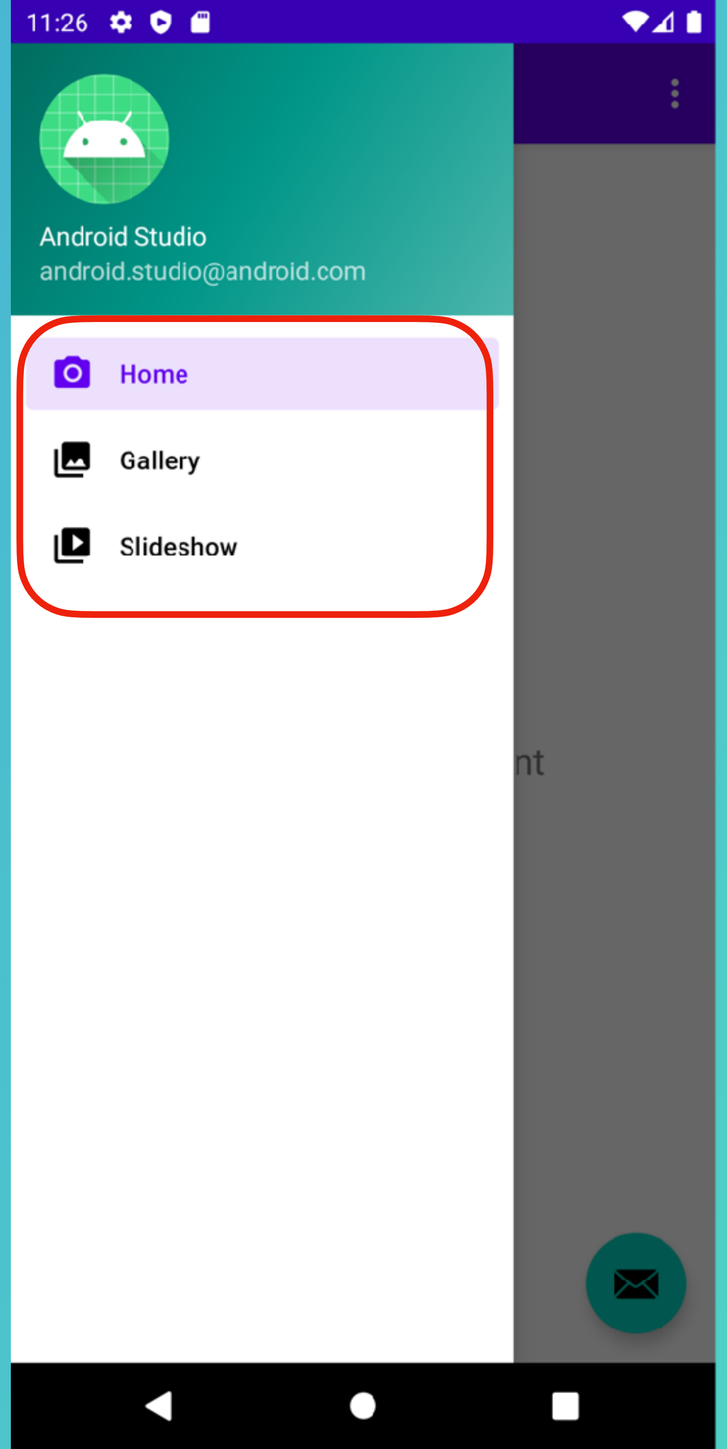
```
buildscript {
   ext.nav_version = “2.5.3”
}
…
dependencies {
  implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"
  implementation "androidx.navigation:navigation-ui-ktx:$nav_version"
}
```

Add a drawer to a layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
  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:id="@+id/drawer_layout"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:fitsSystemWindows="true"
  tools:openDrawer="start">
 <include
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
 <com.google.android.material.navigation.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
</androidx.drawerlayout.widget.DrawerLayout>
```

Declare the menu items

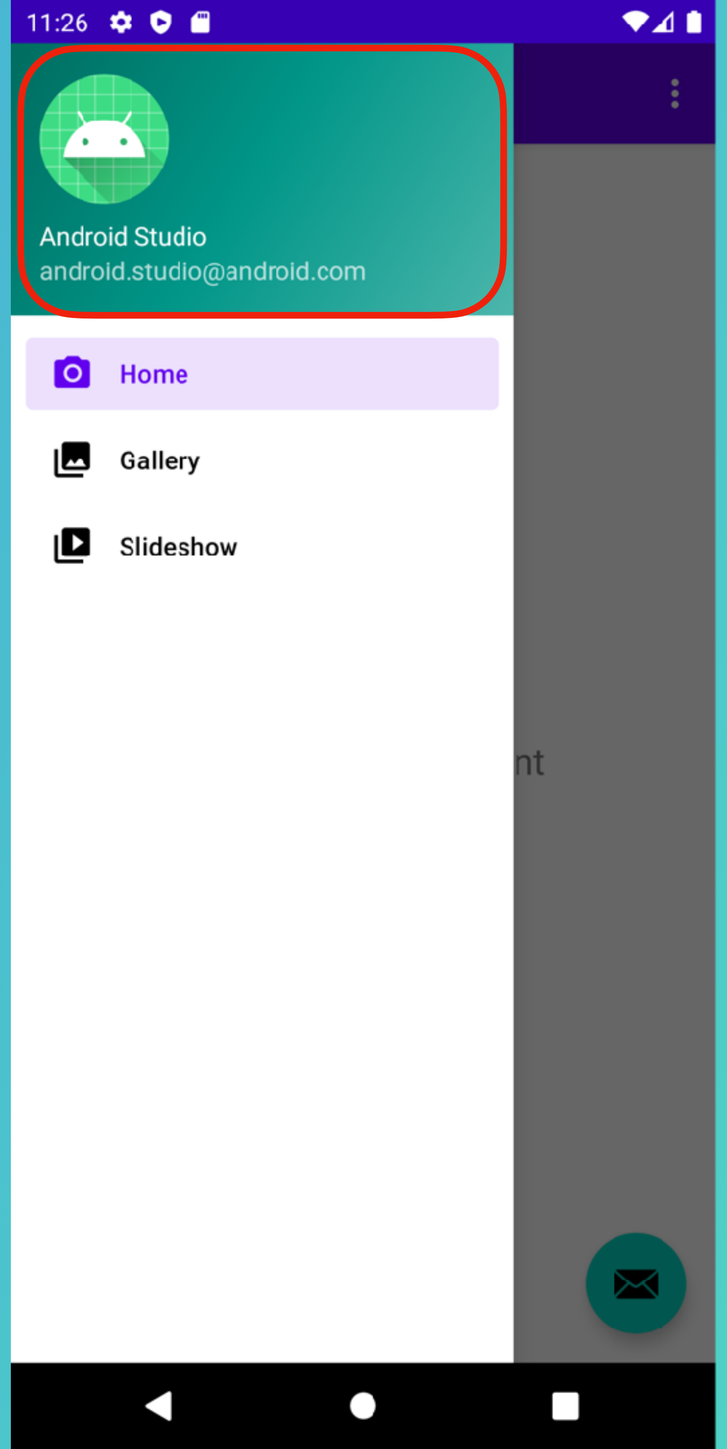
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <group android:checkableBehavior="single">
        <item android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="start"
            android:fitsSystemWindows="true" />
        <item app:menu="@menu/activity_main_drawer" />
            android:id="@+id/nav_gallery"
            android:icon="@drawable/ic_menu_gallery"
            android:title="@string/gallery" />
        <item
            android:id="@+id/nav_slideshow"
            android:icon="@drawable/ic_menu_slideshow"
            android:title="@string/slideshow" />
        ...
    </group>
</menu>
```



Add a header to the nav drawer

```
<android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true" />
    app:headerLayout="@layout/nav_header_main" />
```

```
<?xml version="1.0" encoding="utf-8"?>
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="192dp"
        android:background="?attr/colorPrimaryDark"
        android:padding="16dp"
        android:theme="@style/ThemeOverlay.AppCompat.Dark"
        android:orientation="vertical"
        android:gravity="bottom">

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="My header title"
            android:textAppearance=
              "@style/TextAppearance.AppCompat.Body1"/>
    </LinearLayout>
```



Handle navigation events

```
class MainActivity : AppCompatActivity() {

  private lateinit var appBarConfiguration: AppBarConfiguration

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)

    val navController = findNavController(R.id.nav_host_fragment)

    appBarConfiguration = AppBarConfiguration(
        setOf(
            R.id.nav_home,
            R.id.nav_gallery
        ), drawer_layout
    )
    setupActionBarWithNavController(navController, appBarConfiguration)
    nav_view.setupWithNavController(navController)
  }
}
```

Add a toolbar

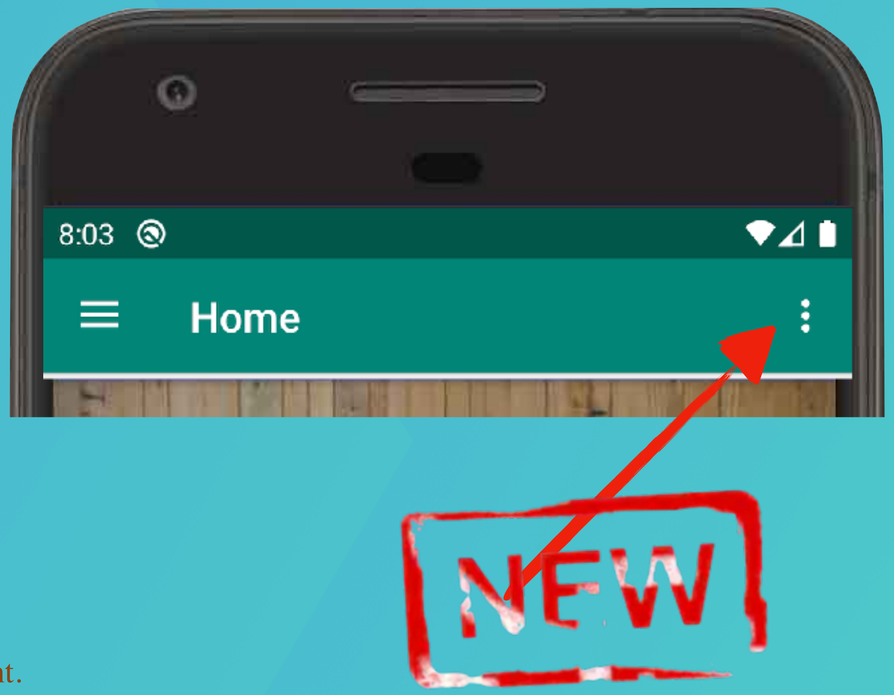
OLD

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.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=".MainActivity">
 <com.google.android.material.appbar.AppBarLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:theme="@style/AppTheme.NoActionBar">
  <androidx.appcompat.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" />
 </com.google.android.material.appbar.AppBarLayout>
```

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(
    class MainActivity : AppCompatActivity() {
        savedInstanceState: Bundle?) {
    override fun onCreate(savedInstanceState: Bundle?) {
        ...
        setSupportActionBar(toolbar)
        ...
    override fun onCreate(savedInstanceState: DrawerLayout
        savedInstanceState: Bundle?) {
        val actionbar = supportActionBar
        actionbar?.apply {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setDisplayHomeAsUpEnabled(true)
    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        setContentView(R.layout.activity_main)
        // Inflate the menu; this adds items to the action bar if it is present.
        menuInflater.inflate(R.menu.menu_main, menu)
        return true
    }
    ...
}
```

```
<manifest>
    <activity
        android:theme="@style/Theme.NoActionBar" >
        android:theme="@style/AppTheme.AppBarOverlay">
        android:name=".SettingsActivity" />
</manifest>
```

```
        val toggle = ActionBarDrawerToggle(
            this, drawer_layout, toolbar,
            R.string.navigation_drawer_open,
            R.string.navigation_drawer_close)
        drawer_layout.addDrawerListener(toggle)
        toggle.syncState()
    }
}
```




```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        ...
        setSupportActionBar(toolbar)
        ...
    }
}
```

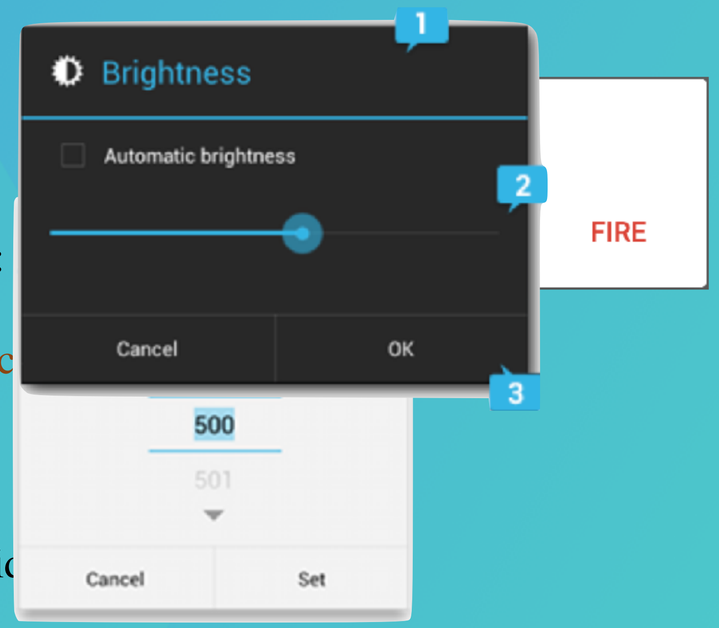
Other State Changes

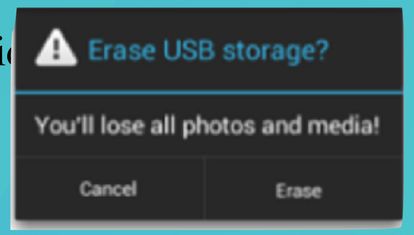

```
drawer_layout.addDrawerListener(
  object : DrawerLayout.DrawerListener {
    override fun onDrawerSlide(drawerView: View, slideOffset: Float) {
      // Respond when the drawer's position changes
    }

    override fun onDrawerOpened(drawerView: View) {
      // Respond when the drawer is opened
    }

    override fun onDrawerClosed(drawerView: View) {
      // Respond when the drawer is closed
    }

    override fun onDrawerStateChanged(newState: Int) {
      // Respond when the drawer motion state changes
    }
  }
)
```

Dialogs

```
class FireMissilesDialogFragment : DialogFragment() {

    override fun onCreateDialog(savedInstanceState: Bundle):
        return activity?.let {
            // Use the Builder class for convenient dialog construc
// 1. Instantiate an AlertDialog.Builder with its constructor
val builder = AlertDialog.Builder(it)
            builder.setMessage(R.string.dialog_fire_
            // Add the buttons
            builder.setPositiveButton(R.string.fire,
                DialogInterface.OnClickListener { dialog, id
                    // FIRE ZE MISSILES!
                })
// 2. Chain together various setter methods to set the dialog characteristics
builder?.setMessage(R.string.dialog_message)
        .setTitle(R.string.dialog_title)
            setNegativeButton(R.string.cancel,
                DialogInterface.OnClickListener { dialog, id
                    // User cancelled the dialog
                })
// 3. Get the AlertDialog from create()
val dialog: AlertDialog? = builder?.create()
            // Create the AlertDialog object and return it
            builder.create()
        } ?: throw IllegalStateException("Activity cannot be null")
    }
}
```

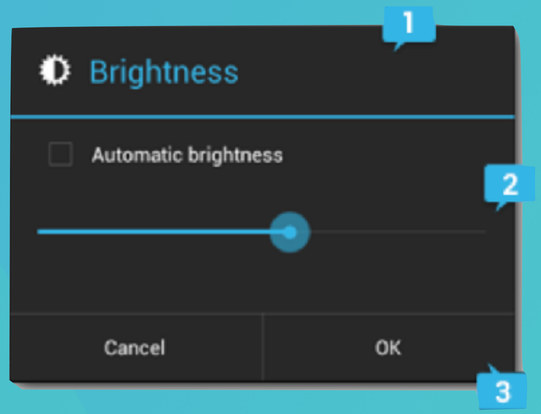
- Small window
- Prompts the user to take a decision
- Modal, by default.





https://developer.android.com/guide/topics/ui/dialogs

Adding actions




```
val alertDialog: AlertDialog? = activity?.let {
    val builder = AlertDialog.Builder(it)
    builder.apply {
        setPositiveButton(R.string.ok,
                DialogInterface.OnClickListener { dialog, id ->
                    // User clicked OK button
                })
        setNegativeButton(R.string.cancel,
                DialogInterface.OnClickListener { dialog, id ->
                    // User cancelled the dialog
                })
    }
    // Set other dialog properties
    ...

    // Create the AlertDialog
    builder.create()
}
```

https://developer.android.com/guide/topics/ui/dialogs

Using Anko

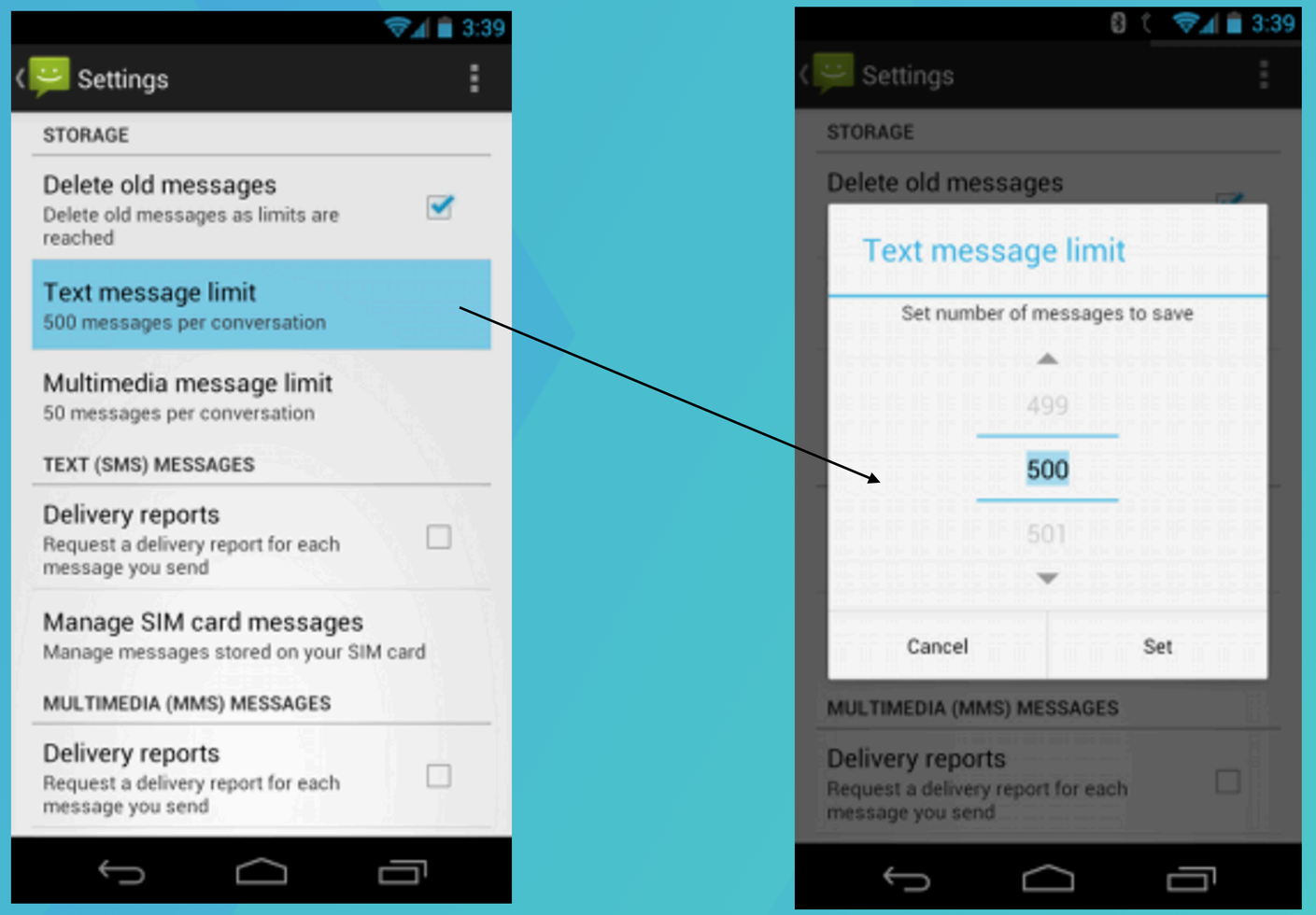

```
alert {
    isCancelable = false
    lateinit var datePicker: DatePicker
    customView {
        verticalLayout {
            datePicker = datePicker {
                maxDate = System.currentTimeMillis()
            }
        }
    }
    yesButton {
        val parsedDate =
          "${datePicker.dayOfMonth}/${datePicker.month + 1}/${datePicker.year}"
        toast("Selected date: $parsedDate")
    }
    noButton { }
}.show()
```

https://github.com/Kotlin/anko/wiki/Anko-Commons---Dialogs

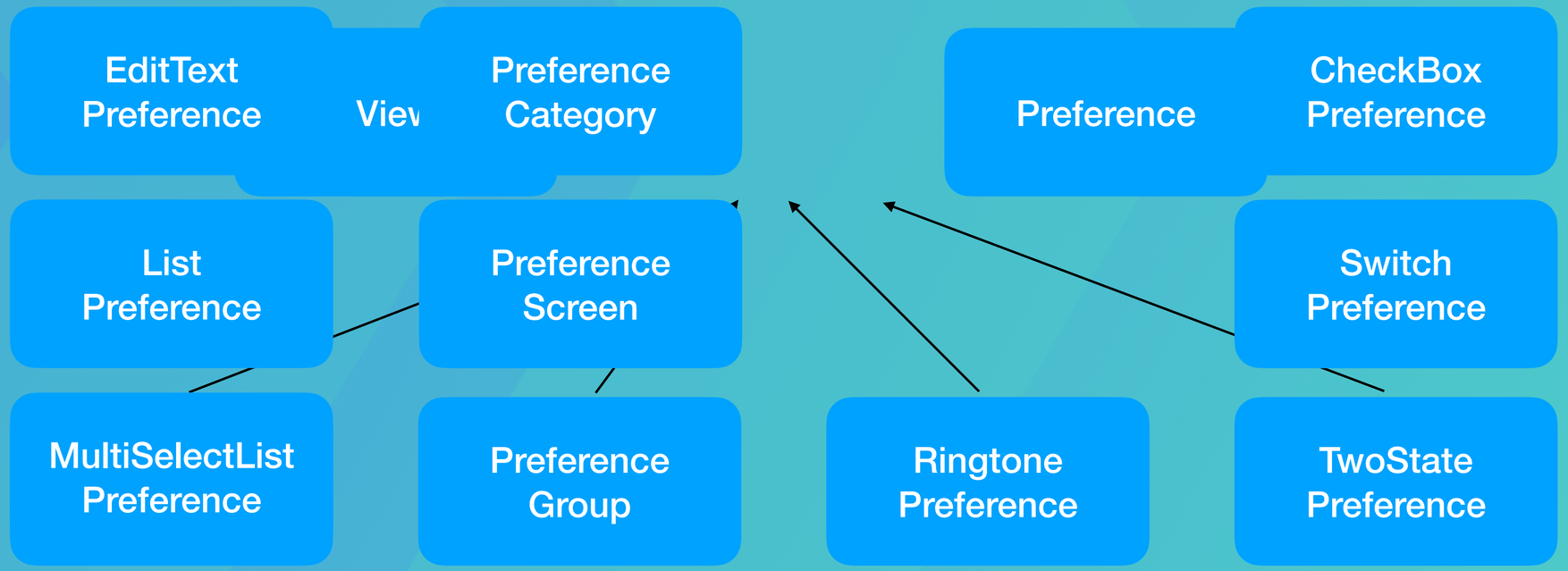
Jetpack Compose

```
AlertDialog(
  onDismissRequest = {
    // Dismiss the dialog when the user clicks outside the dialog or on the back
    // button. If you want to disable that functionality, simply use an empty
    // onCloseRequest.
    openDialog.value = false
  },
  title = { Text(text = "Dialog Title")},
  text = { Text("Here is a text ")},
  confirmButton = {
    Button(
      onClick = { openDialog.value = false }
    ) { Text("This is the Confirm Button")  }
  },
  dismissButton = {
    Button(
      onClick = { openDialog.value = false }
    ) { Text("This is the dismiss Button") }
  }
)
```

Preferences

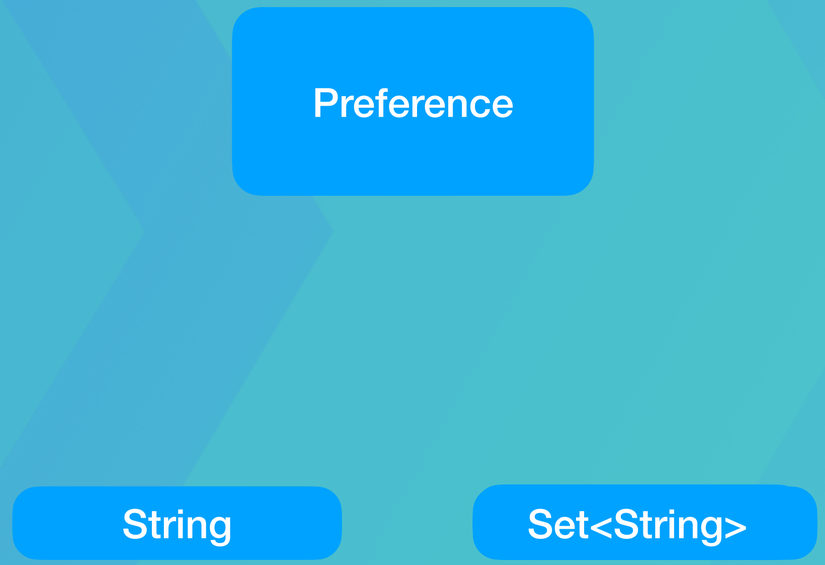


Preferences



https://developer.android.com/reference/android/preference/Preference

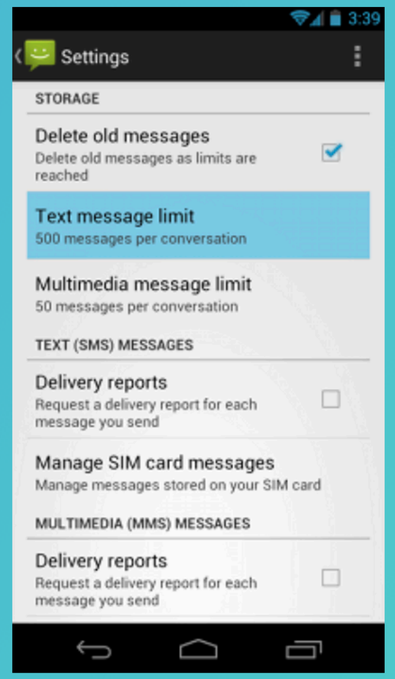
Preferences



PreferenceScreen

DEMO

```xml
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
   <CheckBoxPreference
      android:key="pref_sync"
      android:title="@string/pref_sync"
      android:summary="@string/pref_sync_summ"
      android:defaultValue="true" />
      <ListPreference
      android:dependency="pref_sync"
      android:key="pref_syncConnectionType"
      android:title="@string/pref_syncConnectionType"
      android:dialogTitle="@string/pref_syncConnectionType"
      android:entries="@array/pref_syncConnectionTypes_entries"
      android:entryValues="@array/pref_syncConnectionTypes_values"
      android:defaultValue="@string/pref_syncConnectionTypes_default" />
</PreferenceScreen>
```




```
class SettingsActivity : PreferenceActivity() {
    override fun onCreate(savedInstanceState: Bundle) {
        super.onCreate(savedInstanceState)
        addPreferencesFromResource(R.xml.preferences)
    }
}
```

Jetpack DataStore

Feature	SharedPreferences	Preferences DataStore	Proto DataStore
Async API	✅ (only for reading changed values, via listener)	✅ (via `Flow`)	✅ (via `Flow`)
Synchronous API	✅ (but not safe to call on UI thread)	❌	❌
Safe to call on UI thread	❌*	✅ (work is moved to `Dispatchers.IO` under the hood)	✅ (work is moved to `Dispatchers.IO` under the hood)
Can signal errors	❌	✅	✅
Safe from runtime exceptions	❌**	✅	✅
Has a transactional API with strong consistency guarantees	❌	✅	✅
Handles data migration	❌	✅ (from SharedPreferences)	✅ (from SharedPreferences)
Type safety	❌	❌	✅ with Protocol Buffers

Using DataStore

```
// Preferences DataStore
implementation "androidx.datastore:datastore-preferences:1.0.0"
```

Create the DataStore

```
// with Preferences DataStore
val dataStore: DataStore<Preferences> = context.createDataStore(
    name = "settings"
)
```

Read Data

```
val MY_COUNTER = preferencesKey<Int>("my_counter")
val myCounterFlow: Flow<Int> = dataStore.data
     .map { currentPreferences ->
      currentPreferences[MY_COUNTER] ?: 0
  }
```

Using DataStore

Write Data

```
suspend fun incrementCounter() {
    dataStore.edit { settings ->
        // We can safely increment our counter without losing data due to races!
        val currentCounterValue = settings[MY_COUNTER] ?: 0
        settings[MY_COUNTER] = currentCounterValue + 1
    }
}
```

Using DataStore

Write Data

```
suspend fun incrementCounter() {
    dataStore.edit { settings ->
        // We can safely increment our counter without losing data due to races!
        val currentCounterValue = settings[MY_COUNTER] ?: 0
        settings[MY_COUNTER] = currentCounterValue + 1
    }
}
```

Migrate from SharedPreferences

```
val dataStore: DataStore<Preferences> = context.createDataStore(
    name = "settings",
    migrations = listOf(SharedPreferencesMigration(context, "settings_preferences"))
)
```

https://developer.android.com/topic/libraries/architecture/datastore

Saving & Reading Local Files

- Internal storage
  - Internal cache files
- External storage
- Shared preferences
- Databases

https://developer.android.com/guide/topics/data

Internal Storage

- It's always available.
- Available only to your app.
- On uninstall everything is removed.

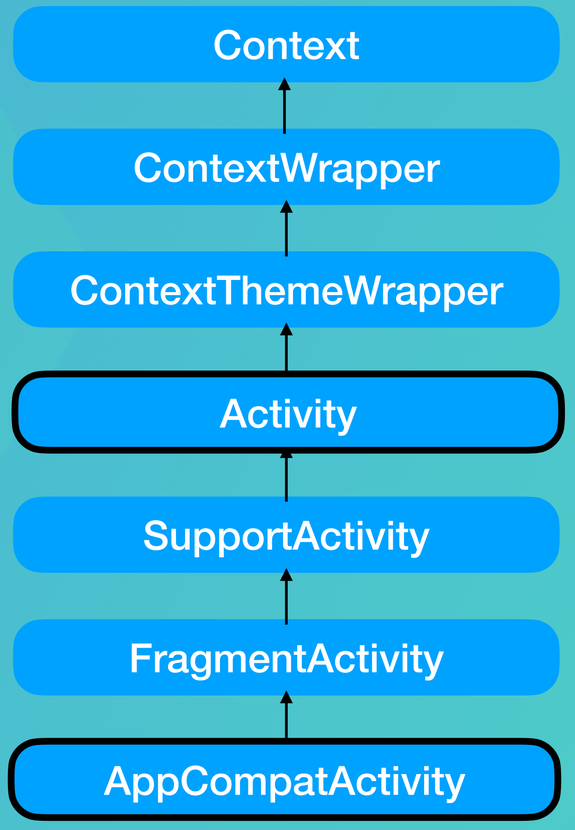
Neither the user nor other apps can access your files!

https://developer.android.com/guide/topics/data/data-storage#filesInternal

Internal Storage


```
val file = File(context.filesDir, filename)

val filename = "myfile"
val fileContents = "Hello world!"
context.openFileOutput(filename,
   Context.MODE_PRIVATE).use {
     it.write(fileContents.toByteArray())
}

private fun getTempFile(
    context: Context, url: String): File? =
 Uri.parse(url)?.lastPathSegment?.let { filename ->
    File.createTempFile(filename, null, context.cacheDir)
 }
```

```
public abstract class Context {
    ...
    public abstract File getFilesDir();
    ...
    public abstract File getCacheDir();
    ...
}
```



External Storage Permissions

```
<manifest ...>
    <uses-permission android:name=
      "android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name=
      "android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

< Android 4.4 (API level 19)


```
/* Checks if external storage is available for read and write */
fun isExternalStorageWritable(): Boolean {
    return Environment.getExternalStorageState() == Environment.MEDIA_MOUNTED
}
/* Checks if external storage is available to at least read */
fun isExternalStorageReadable(): Boolean {
     return Environment.getExternalStorageState() in
        setOf(Environment.MEDIA_MOUNTED, Environment.MEDIA_MOUNTED_READ_ONLY)
}
```

https://developer.android.com/training/data-storage/files#WriteInternalStorage

FileProvider

content://com.example.myapp.fileprovider/myimages/default_image.jpg

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">
    <application
        ...>
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.example.myapp.fileprovider"
            android:grantUriPermissions="true"
            android:exported="false">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
    </application>
</manifest>
```

```
<paths>
    <files-path path="images/" name="myimages" />
</paths>
```

https://developer.android.com/training/secure-file-sharing/setup-sharing

Lecture outcomes

- Navigate between screens/views.
- Use dialogs and pickers.
- Manage files & preferences.