Navigation Graph & Data Binding

Muhammad Haris Bin Abid

Navigation component

- Collection of libraries and tooling, including an integrated editor, for creating navigation paths through an app
- Assumes one `Activity` per graph with many `Fragment` destinations
- Consists of three major parts:
  - Navigation graph
  - Navigation Host (`NavHost`)
  - Navigation Controller (`NavController`)

Add dependencies

In build.gradle, under dependencies:

```
implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"
implementation "androidx.navigation:navigation-ui-ktx:$nav_version"

//if using libs.version.toml
[versions]
# Your existing versions
nav = "2.7.6" # Use the latest stable version

[libraries]
# Your existing libraries
androidx-navigation-fragment-ktx = { group = "androidx.navigation", name = "navigation-fragment-ktx", version.ref = "nav" }
androidx-navigation-ui-ktx = { group = "androidx.navigation", name = "navigation-ui-ktx", version.ref = "nav" }

// gradle
implementation(libs.androidx.navigation.fragment.ktx)
implementation(libs.androidx.navigation.ui.ktx)
```

Navigation host (NavHost)

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/nav_host"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost="true"
    app:navGraph="@navigation/nav_graph_name"/>
```

Navigation graph

New resource type located in res/navigation directory

- XML file containing all of your navigation destinations and actions
- Lists all the (Fragment/Activity) destinations that can be navigated to
- Lists the associated actions to traverse between them
- Optionally lists animations for entering or exiting

Creating a Fragment

- Extend `Fragment` class
- **Override** onCreateView()
- Inflate a layout for the Fragment that you have defined in XML

```
class DetailFragment : Fragment() {
  override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
    return inflater.inflate(R.layout.detail_fragment, container, false)
  }
}
```

Specifying Fragment destinations

- Fragment destinations are denoted by the `action` tag in the navigation graph.
- Actions can be defined in XML directly or in the Navigation Editor by dragging from source to destination.
- Autogenerated action IDs take the form of action_<sourceFragment>_to_<destinationFragment>.

Example fragment destination

```
<fragment
    android:id="@+id/welcomeFragment"
    android:name="com.example.android.navigation.WelcomeFragment"
    android:label="fragment_welcome"
    tools:layout="@layout/fragment_welcome" >

    <action
        android:id="@+id/action_welcomeFragment_to_detailFragment"
        app:destination="@id/detailFragment" />
</fragment>
```

Navigation Controller (NavController)

`NavController` manages UI navigation in a navigation host.

- Specifying a destination path only names the action, but it doesn't execute it.
- To follow a path, use `NavController`.

Example NavController

```
class MainActivity : AppCompatActivity() {

  override fun onCreate(savedInstanceState: Bundle?) {

    ...

val navHostFragment = supportFragmentManager.findFragmentById(R.id.nav_host) as NavHostFragment

val navController = navHostFragment.navController

  }

  fun navigateToDetail() {

    navController.navigate(R.id.action_welcomeFragment_to_detailFragment)

  }

}
```

Task

A task is a collection of activities that users interact with.

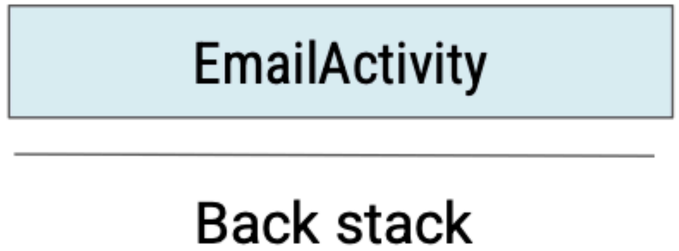
Key characteristics of a task:

- A task is essentially a stack of activities
- Each task can exist independently of other tasks
- Tasks can be moved to the background and foreground
- By default, all activities in an app belong to the same task

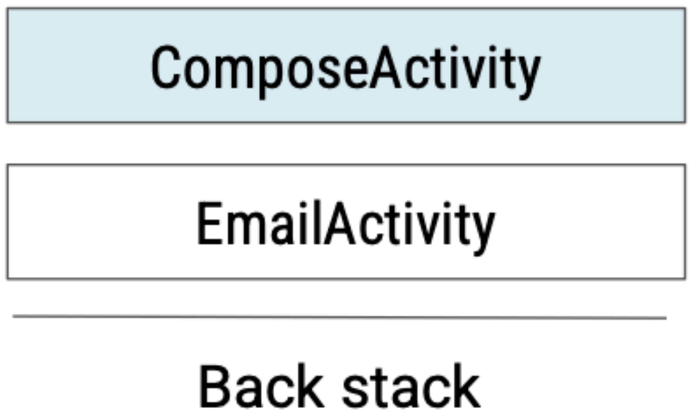
For more info : https://developer.android.com/guide/components/activities/tasks-and-back-stack

Back stack of activities

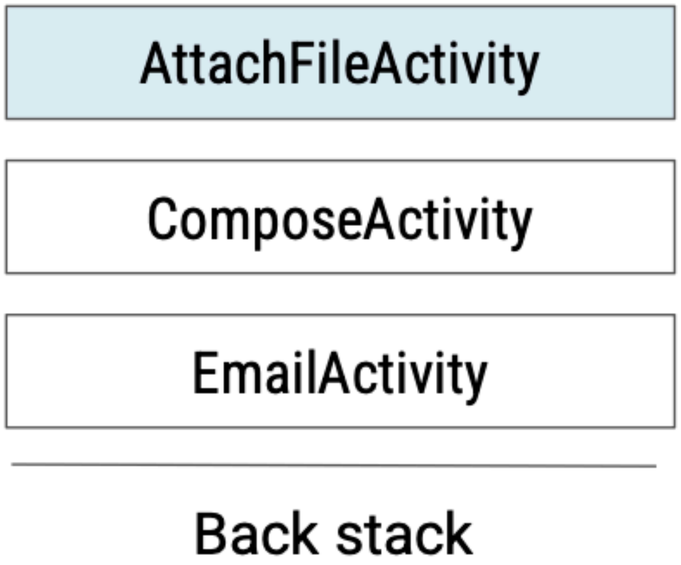
The back stack refers specifically to the ordered arrangement of activities within a task.



Add to the back stack

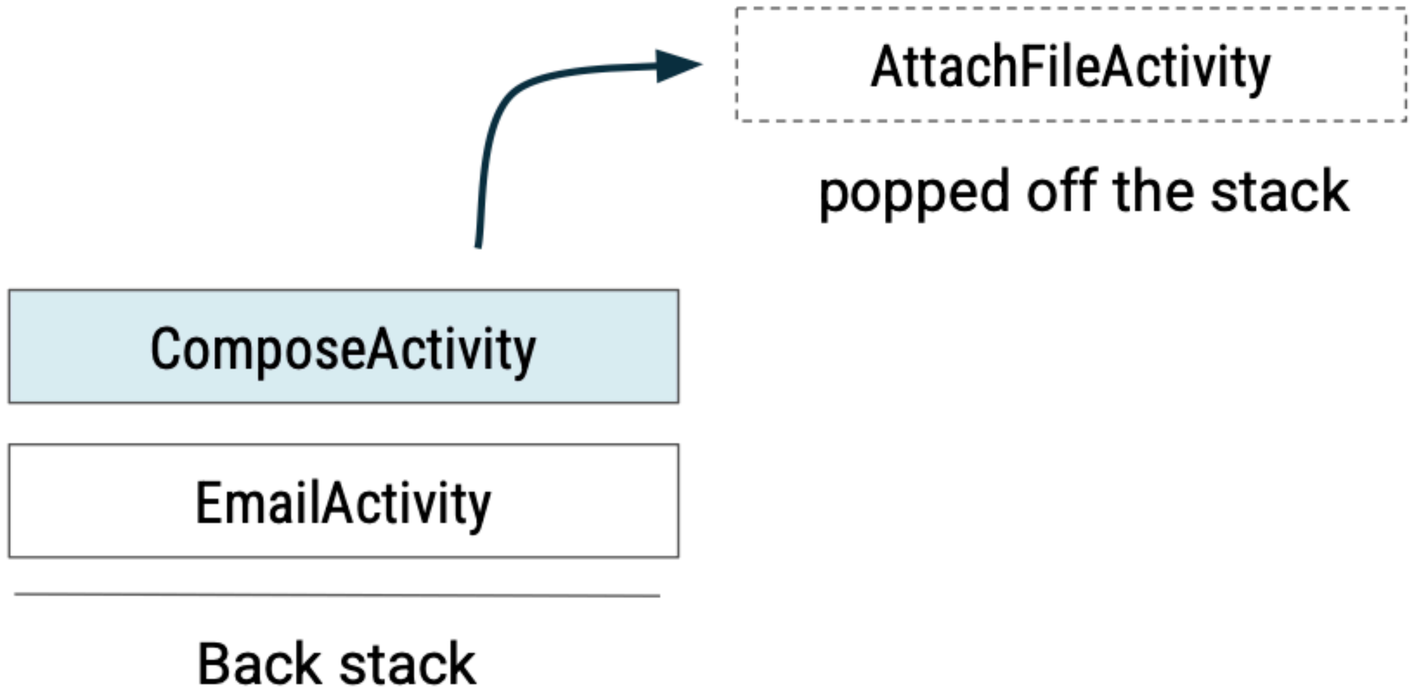


Add to Back stack again

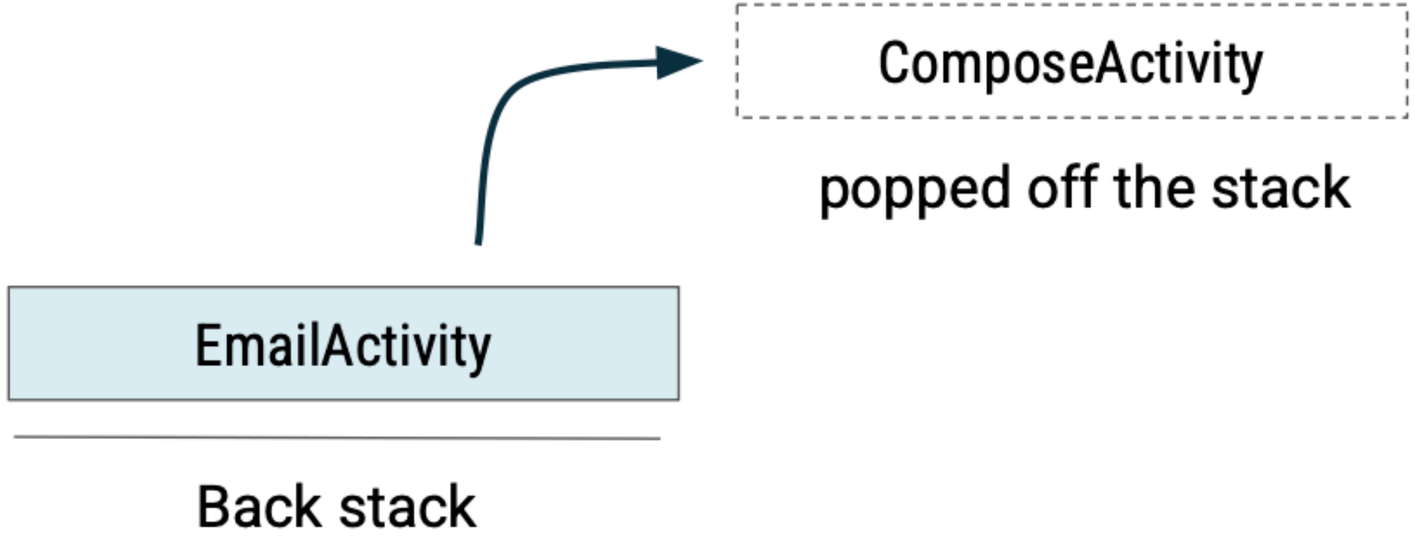


Tap Back button

The back stack refers specifically to the ordered arrangement of activities within a task.

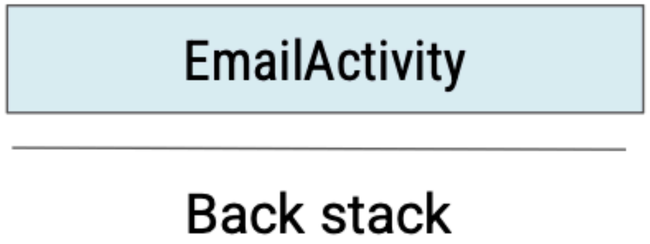


Tap Back button again



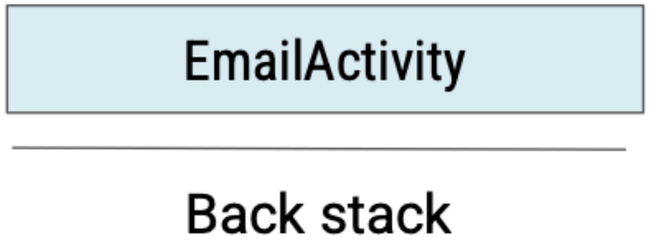
Back stack of activities

The back stack refers specifically to the ordered arrangement of activities within a task.



Back stack of activities

The back stack refers specifically to the ordered arrangement of activities within a task.



Data Binding

Data Binding

Current approach: findViewById()

Traverses the `View` hierarchy each time

MainActivity.kt

```
val name = findViewById(...)
val age = findViewById(...)
val loc = findViewById(...)

name.text = …
age.text = …
loc.text = …
```

findViewById →
←
findViewById →
←
findViewById →
←

activity_main.xml

```
<ConstraintLayout … >
  <TextView
      android:id="@+id/name"/>
  <TextView
      android:id="@+id/age"/>
  <TextView
      android:id="@+id/loc"/>
</ConstraintLayout>
```

Use data binding instead

Bind UI components in your layouts to data sources in your app.

Bind UI components in your layouts to data sources in your app.

MainActivity.kt

```
Val binding:ActivityMainBinding

binding.name.text = …
binding.age.text = …
binding.loc.text = …
```

initialize binding

activity_main.xml

```
<layout>
   <ConstraintLayout … >
       <TextView
          android:id="@+id/name"/>
       <TextView
          android:id="@+id/age"/>
       <TextView
          android:id="@+id/loc"/>
   </ConstraintLayout>
</layout>
```

Modify build.gradle file

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-kapt' // Add this line
}

android {
    ...

    buildFeatures {
        dataBinding = true
    }

    ...
}
```

Add layout tag

```
<layout>
    <androidx.constraintlayout.widget.ConstraintLayout>
        <TextView ... android:id="@+id/username" />
        <EditText ... android:id="@+id/password" />
    </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

Layout inflation with data binding

Replace this

```
setContentView(R.layout.activity_main)
```

with this

```
val binding: ActivityMainBinding = DataBindingUtil.setContentView(
    this, R.layout.activity_main)
binding.username = "Melissa"
```

Data binding layout variables

```
<layout>
 <data>
    <variable name="name" type="String"/>
  </data>
  <androidx.constraintlayout.widget.ConstraintLayout>
    <TextView
      android:id="@+id/textView"
      android:text="@{name}" />
  </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

In MainActivity.kt:

```
binding.name = "John"
```

Data binding layout expressions

```
<layout>
  <data>
    <variable name="name" type="String"/>
  </data>

  <androidx.constraintlayout.widget.ConstraintLayout>
    <TextView
      android:id="@+id/textView"
      android:text="@{name.toUpperCase()}" />
  </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

Thank you