AND205

Android Navigation

Download class materials from
university.xamarin.com

Microsoft

Xamarin University

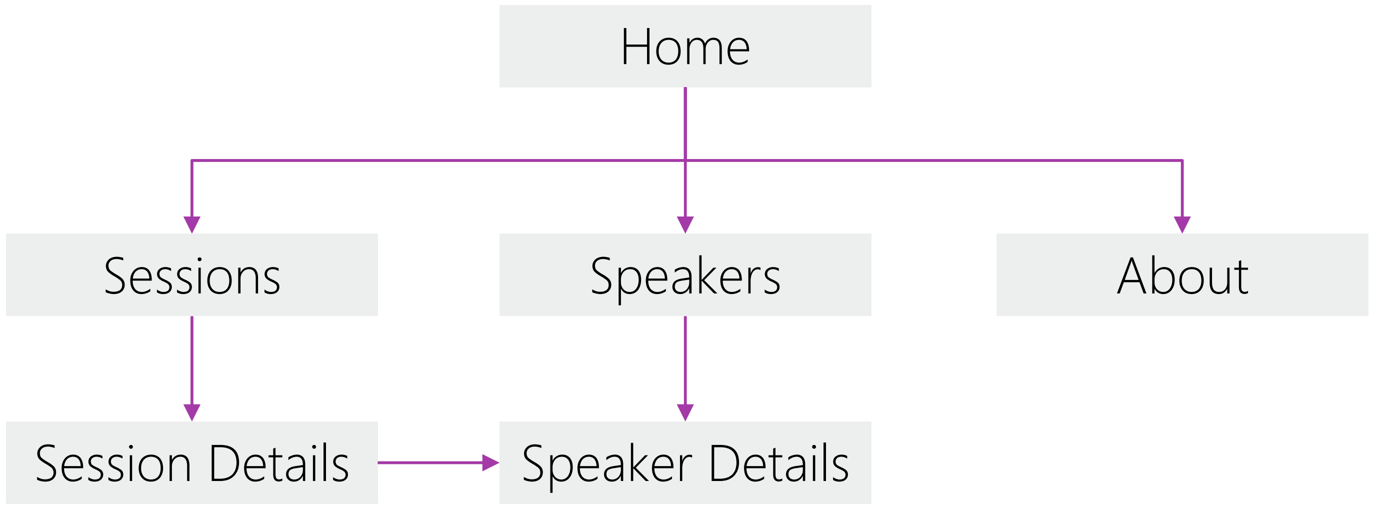
Objectives

- Use `Fragment` and `FrameLayout` to swap views
- Implement tab navigation with `TabLayout`
- Implement gestural navigation with `ViewPager`
- Combine gestural and tab navigation
- Implement drawer navigation with `DrawerLayout`

Xamarin University

What is navigation?

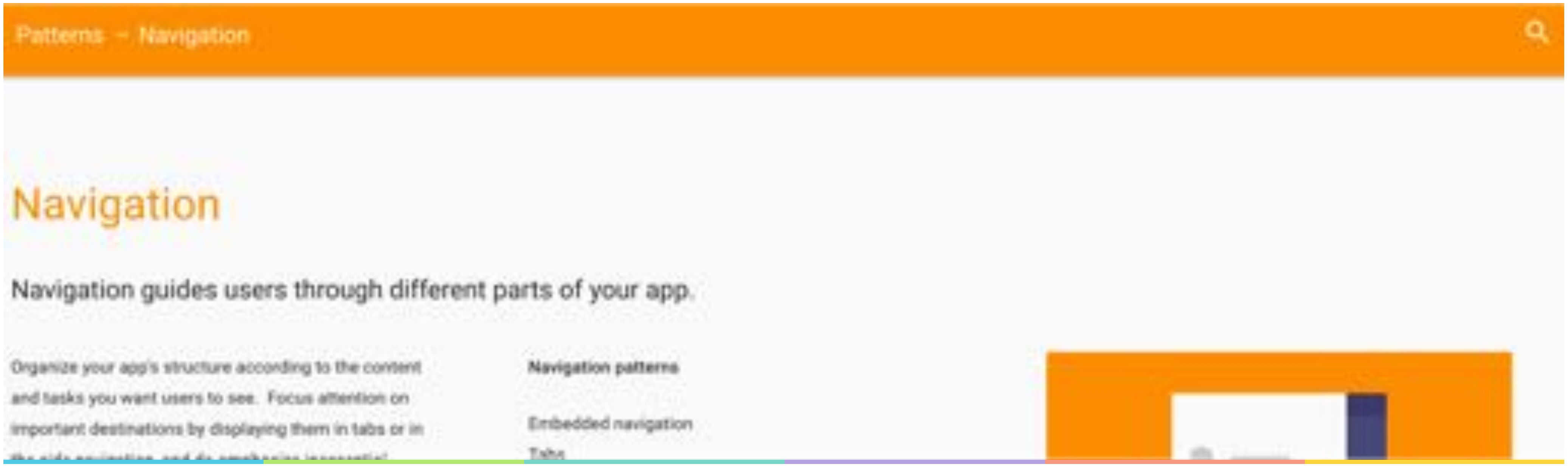
- *Navigation* is the set of transitions between the parts of your app; the term is typically understood to include the UI and user actions needed to make the transitions



Platform paradigms

- You should follow Google's navigation guidelines since they will be familiar to users from their experience with other Android apps

Source: material.google.com



Patterns → Navigation

Navigation

Navigation guides users through different parts of your app.

Organize your app's structure according to the content and tasks you want users to see. Focus attention on important destinations by displaying them in tabs or in

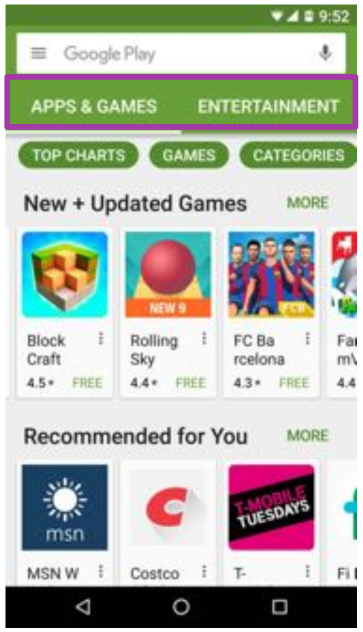
Navigation patterns

Embedded navigation

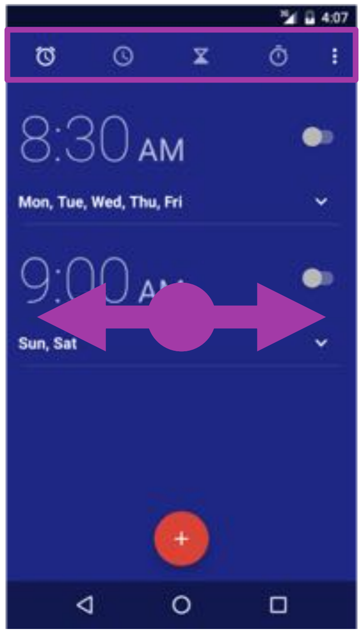
Xamarin University

Discussion

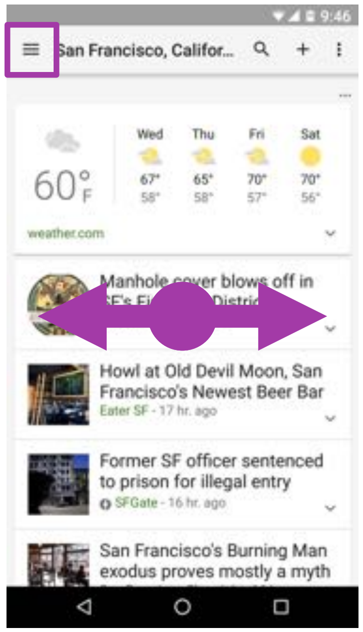
- Which top-level navigation paradigm(s) are used by these apps?



Tab



Tab and Gestural


9:46
San Francisco, Califor...
60°F
Wed
Thu
Fri
Sat
weather.com
Manhole cover blows off in SF's Fi... District
Howl at Old Devil Moon, San Francisco's Newest Beer Bar
Eater SF - 17 hr. ago
Former SF officer sentenced to prison for illegal entry
SFGate - 16 hr. ago
San Francisco's Burning Man exodus proves mostly a myth

Drawer and Gestural

Use Fragment and FrameLayout to swap views

Xamarin University

Tasks

1. Add a `Fragment` to a `FrameLayout` dynamically

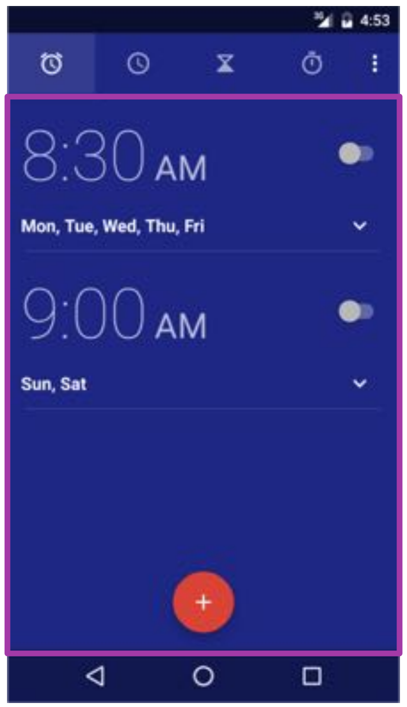
Xamarin University

Motivation

❖ Activities are too large to be the core building blocks of a dynamic UI

Want to keep this part the same...

...while replacing the content here

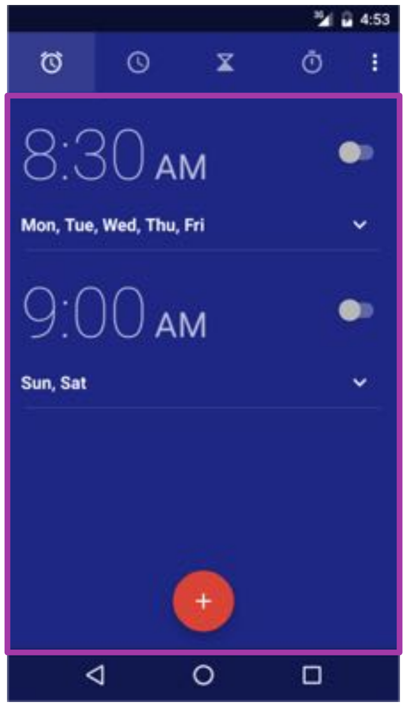


Xamarin University

What is FrameLayout?

- A **`FrameLayout`** is a container that is intended to hold a single child, it is common to set the child from code

Use a **`FrameLayout`** here and replace its child dynamically →



← The child can be any View, including a Fragment

Using FrameLayout

❖ `FrameLayout` methods let you update its child view

No children in the XML →

```
<FrameLayout android:id ="@+id/myFrame" ... />
```

Remove old child →

Add new child →

```
void ShowInFrame(string message)
{
   var frame = FindViewById<FrameLayout>(Resource.Id.myFrame);

   if (frame.ChildCount > 0)
      frame.RemoveViewAt(0);

   var tv = new TextView(this) { Text = message };
   frame.AddView(tv);
}
```

Xamarin University

What is a Fragment?

- A *Fragment* is a unit of UI + behavior intended for use with dynamic UI

MyFragment.axml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout ... >
   ...
</LinearLayout>
```

↑

UI

MyFragment.cs

```
public class MyFragment : Fragment
{
   ...
}
```

↑

Behavior

Xamarin University

Standard vs. support Fragments

❖ Android provides two Fragment implementations: standard and support

Standard

```
namespace Android.App
{  ...
   public class Fragment : ... { ... }
}
```

Support

```
namespace Android.Support.V4.App
{  ...
   public class Fragment : ... { ... }
}
```

We'll use support Fragments because other types we use require them (e.g. `ViewPager`).

Xamarin University

Fragment types

❖ The support library provides four types that help you work with Fragments inside your Activities

```
public class Fragment            : ... { ... }
public class FragmentActivity    : ... { ... }
public class FragmentManager     : ... { ... }
public class FragmentTransaction : ... { ... }
```

You will use all of these types

Xamarin University

What is Fragment?

- **`Fragment`** is the base type for all of your Fragments – it defines the lifecycle methods

Inherit from **`Fragment`**

```
public class MyFragment : Android.Support.V4.App.Fragment
{  ...
   public override View OnCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
   {
      var view = inflater.Inflate(Resource.Layout.MyLayoutFile, container, false);

      var tv = view.FindViewById<TextView>(Resource.Id.myTextView);
      ...
      return view;
   }
}
```

Create your UI from a layout file

Work with the UI elements in your layout (if needed)

Xamarin University

What is FragmentActivity?

- **`FragmentActivity`** is the base type for your Activities – it adds properties to let your Activity host support Fragments

If you are using support Fragments, use this as your base class...

```
public class FragmentActivity : ...
{  ...
   public virtual Android.Support.V4.App.FragmentManager SupportFragmentManager { get; }
}
```

... and this property to work with Fragments

The standard Activity class (i.e. non-support) has an analogous property named **`FragmentManager`** that returns the standard version of the **`FragmentManager`** type.

What is FragmentManager?

- **FragmentManager** helps you dynamically add/remove fragments from your Activity's UI

```
public class FragmentManager : ...
{  ...
   public abstract FragmentTransaction BeginTransaction();
}
```

All changes to your Activity's fragments are done through the manager, it is your source for Fragment transactions

What is FragmentTransaction?

❖ **`FragmentTransaction`** swaps the fragments your Activity displays (Android requires these fragment changes be done inside a transaction)

```
public abstract class FragmentTransaction
{  ...
   public abstract FragmentTransaction Remove (Fragment fragment);
   public abstract FragmentTransaction Add    (int containerViewId, Fragment fragment);
   public abstract FragmentTransaction Replace(int containerViewId, Fragment fragment);

   public abstract int Commit();
}
```

It does the add/remove from your container for you (the container will typically be a **`FrameLayout`**)

How to replace a fragment

- **`FragmentTransaction`** handles the details of loading a new fragment into your UI

```
public class MainActivity : Android.Support.V4.App.FragmentActivity
{  ...
   void ShowFragment()
   {
      var fragment = new MyFragment();

      var transaction = base.SupportFragmentManager.BeginTransaction();
      transaction.Replace(Resource.Id.myFrame, fragment);
      transaction.Commit();
  }
}
```

Your **`FrameLayout`**

Your Fragment

Individual Exercise

Use Fragments and FrameLayout to swap views

Xamarin University

Summary

1. Add a **`Fragment`** to a **`FrameLayout`** dynamically

Implement tab navigation with TabLayout

Xamarin University

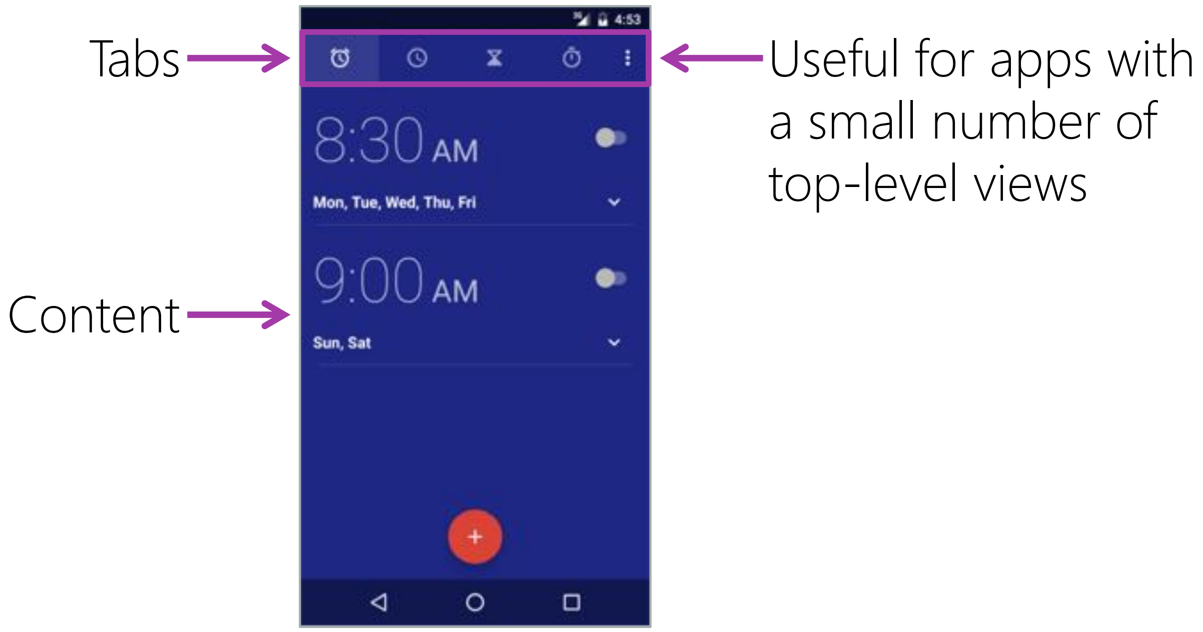
Tasks

1. Include a **`TabLayout`** in your UI
2. Add tabs using code-behind
3. Add tabs using XML
4. Respond when a tab is tapped

Xamarin University

What is tab navigation?

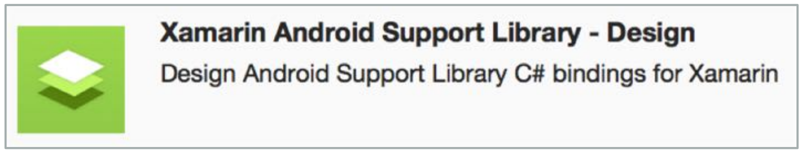
- *Tab navigation* is a navigation paradigm that uses a horizontal row of *tabs* to let the user change the view in an associated content area



Xamarin University

Which types to use for tabs?

❖ Google currently recommends that you use the types from the *Design Support Library* to implement tab navigation



Xamarin Android Support Library - Design
Design Android Support Library C# bindings for Xamarin

1. Add the NuGet package to your project

`AndroidManifest.xml`

```
<application
   ...
   android:theme="@style/Theme.AppCompat">
</application>
```

2. Use an AppCompat theme (required by the Design Support Lib)

Tab classes

- You will use several Design Support Library classes to implement tab navigation

`TabLayout` displays tabs

`TabLayout.Tab` represents a tab

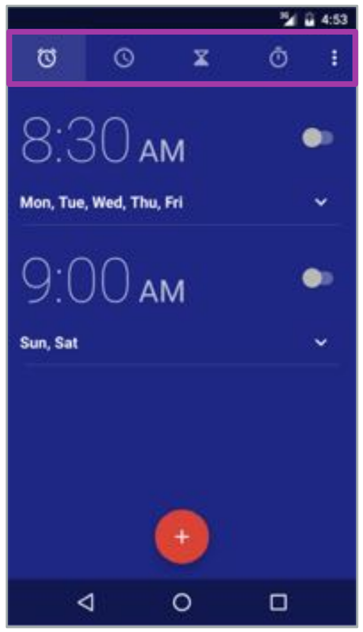
`TabItem` tab proxy for XML

Xamarin University

What is TabLayout?

- `TabLayout` is a layout that hosts a horizontal strip of tabs

`TabLayout` displays tabs



Xamarin University

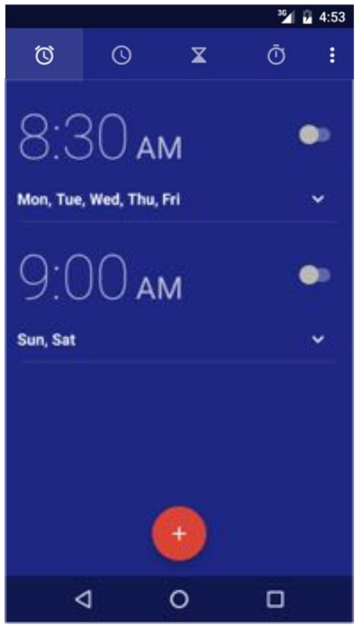
How to use TabLayout

❖ Typically, you will create a **`TabLayout`** in your XML layout file

`TabLayout` only displays the tabs

You will need a separate content area

```
<LinearLayout ...>

   <android.support.design.widget.TabLayout
      android:id="@+id/tabLayout"
      android:layout_height="wrap_content"
      android:layout_width="match_parent" />

   <FrameLayout
      android:id="@+id/contentFrame"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" />

</LinearLayout>
```



What is TabLayout.Tab?

- `TabLayout.Tab` represents a tab

Text → Icon → Custom view → Read-only properties →

```
public sealed class Tab : Java.Lang.Object
{  ...
   public TabLayout.Tab SetText(int      resId) { ... }
   public TabLayout.Tab SetText(string   text ) { ... }

   public TabLayout.Tab SetIcon(int      resId) { ... }
   public TabLayout.Tab SetIcon(Drawable icon ) { ... }

   public TabLayout.Tab SetCustomView(int resId) { ... }
   public TabLayout.Tab SetCustomView(View view) { ... }

   public string   Text       { get; }
   public Drawable Icon       { get; }
   public View     CustomView { get; }
   public int      Position   { get; }
}
```

Xamarin University

How to add tabs in code

❖ To create tabs in code, you must use a factory method from **`TabLayout`**

Create →

Set properties →

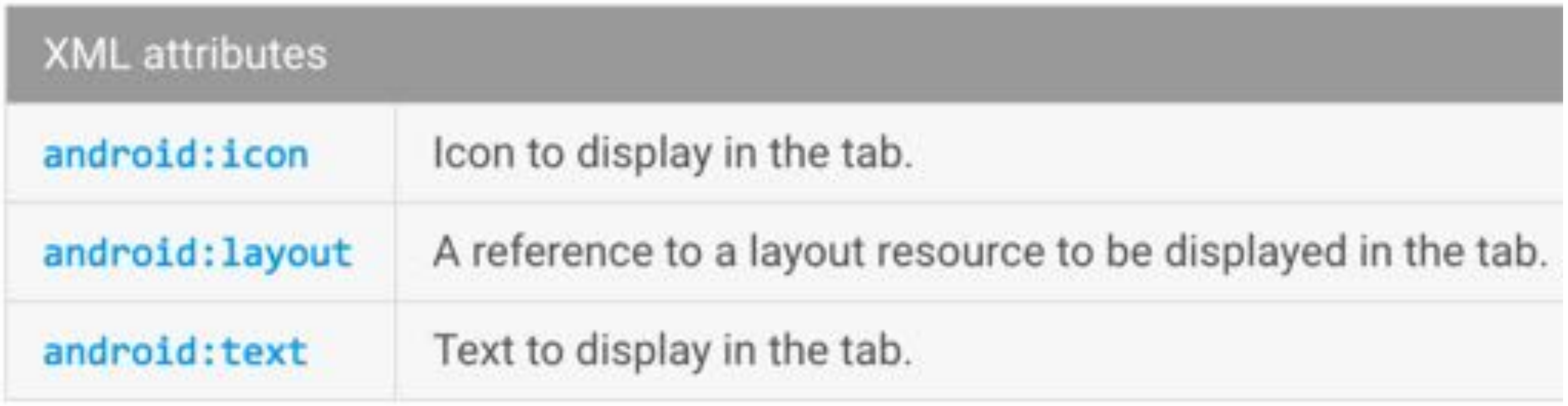
Add to layout →

```
var tabLayout = FindViewById<TabLayout>(Resource.Id.tabLayout);

var tab = tabLayout.NewTab();

tab.SetText("Sessions");
tab.SetIcon(Resource.Drawable.sessions);

tabLayout.AddTab(tab);
```

Xamarin University

What is TabItem?

- A **`TabItem`** is a proxy for a **`TabLayout.Tab`** for use in XML



XML attributes	
`android:icon`	Icon to display in the tab.
`android:layout`	A reference to a layout resource to be displayed in the tab.
`android:text`	Text to display in the tab.

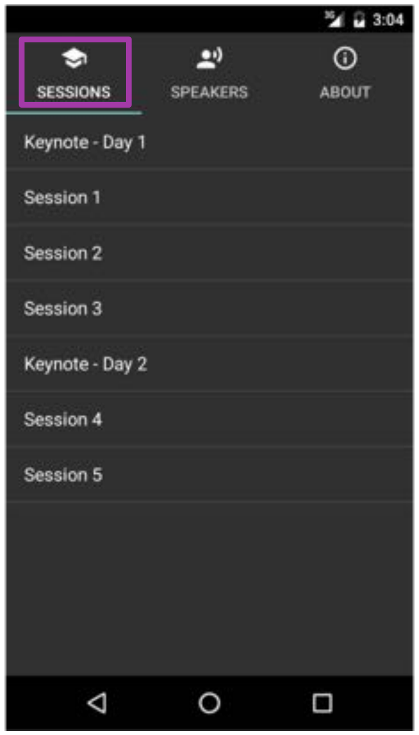
Properties you set in XML control the tab's contents

Xamarin University

How to add tabs in XML

❖ You add a **`TabItem`** to your **`TabLayout`** in XML and it creates a **`TabLayout.Tab`** for you

```
<android.support.design.widget.TabLayout ...>

   <android.support.design.widget.TabItem
      android:text="Sessions"
      android:icon="@drawable/sessions" />
   ...

</android.support.design.widget.TabLayout>
```



Xamarin University

Selection notification

- `TabLayout` has a `TabSelected` event

```
tabLayout.TabSelected += OnTabSelected;
```

```
void OnTabSelected(object sender, TabLayout.TabSelectedEventArgs e)
{
   int position = e.Tab.Position;
   ...
}
```

Typically you would change the Fragment in your content area

Use the tab's position to determine which tab was tapped

Individual Exercise

Implement tab navigation with TabLayout

Summary

1. Include a `TabLayout` in your UI
2. Add tabs using code-behind
3. Add tabs using XML
4. Respond when a tab is tapped

Implement gestural navigation with ViewPager

Xamarin University

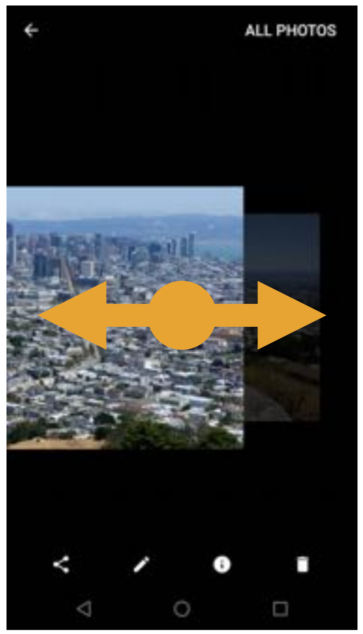
Tasks

1. Add a **`ViewPager`** to your UI
2. Code an adapter to supply the **`ViewPager`** with Fragments

What is gestural navigation?

❖ *Gestural* navigation lets the user switch views using a swipe gesture

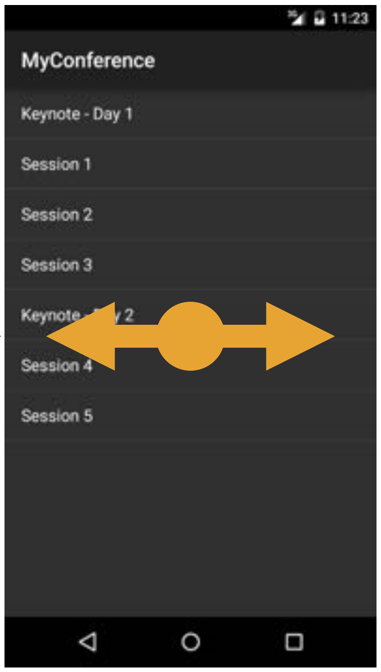
Photos app uses horizontal swipe to navigate between images



What is ViewPager?

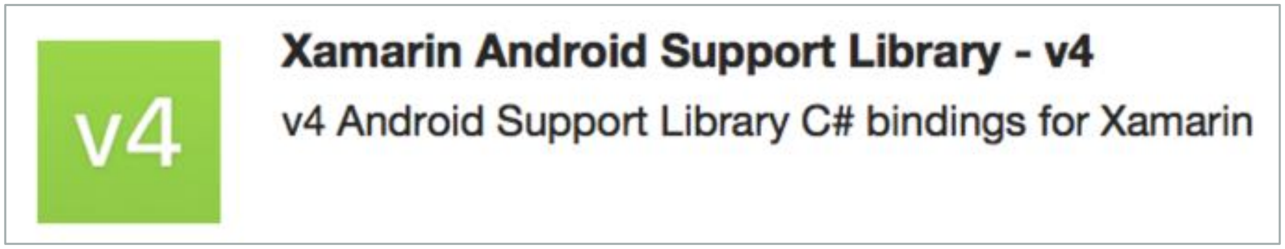
❖ `ViewPager` is a layout manager that implements gestural navigation

Detects the swipe gesture and changes the page for you



Support Library

- `ViewPager` is in the v4 Support Library



Xamarin Android Support Library - v4

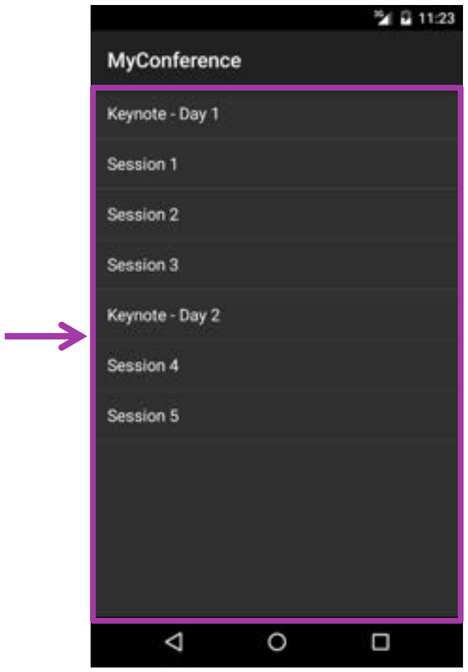
v4 Android Support Library C# bindings for Xamarin

You must include
this in your project

ViewPager content area

- **`ViewPager`** inherits from **`ViewGroup`** so it has an area to display your content

Your pages are hosted by the **`ViewPager`** itself, no need to declare a separate **`FrameLayout`**



Xamarin University

How to use ViewPager

❖ Add a **`ViewPager`** to your layout file

`ViewPager` can be the root node in your XML →

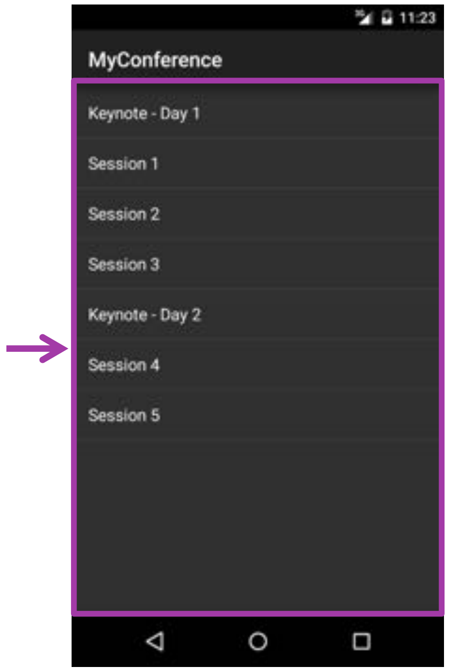
```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.view.ViewPager
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:id="@+id/viewPager"
   android:layout_width="match_parent"
   android:layout_height="match_parent" />
```

↑ Pages are loaded via code-behind, not hardcoded in XML so you can often use a self-closing tag

What are pages?

- The pages displayed by **`ViewPager`** are typically either Fragments or Views (we will use Fragments as they are more powerful and common)

Typically a Fragment



Fragment transactions

- `ViewPager` performs the fragment transactions for you, but you need to supply it with a `FragmentManager`

```
var fragment = new MyFragment();

var transaction = base.SupportFragmentManager.BeginTransaction();
transaction.Replace(Resource.Id.myFrame, fragment);
transaction.Commit();
```

`ViewPager` manages the Fragments for you, you do not need to write this code

Xamarin University

Fragment base type

❖ Fragments displayed by **`ViewPager`** must use the support-library **`Fragment`** class as their base

```
public class MyFragment : Android.Support.V4.App.Fragment
{
    ...
}
```

Required because **`ViewPager`** uses the support version of **`FragmentManager`** for its fragment transactions

Activity base type

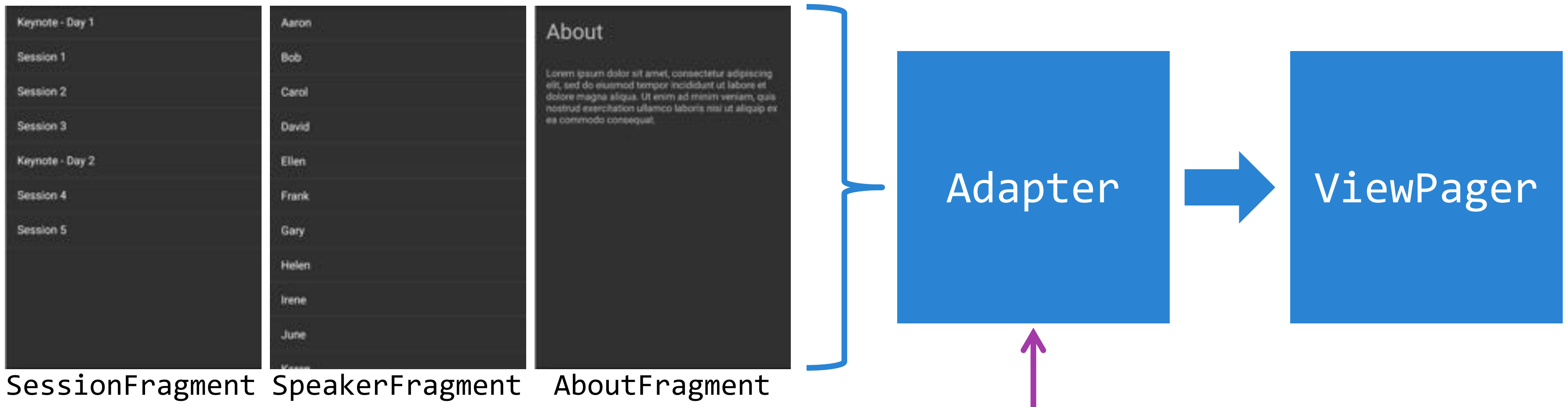
- Activities that host a **`ViewPager`** use the support-library **`FragmentActivity`** class as their base

```
public class MainActivity : Android.Support.V4.App.FragmentActivity
{
    ...
}
```

You inherit a **`SupportFragmentManager`** property that gives you the support version of the **`FragmentManager`** which **`ViewPager`** needs

What is an adapter?

❖ An *adapter* provides your pages to the **`ViewPager`**



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Creates Fragments and gives them to the **`ViewPager`** when requested

Adapter base class

- You code an adapter that inherits from **`FragmentPagerAdapter`**

```
     Android.Support.V4.App.PagerAdapter
                      |
Android.Support.V4.App.FragmentPagerAdapter
                      |
                  MyAdapter
```

Adapter FragmentManager

❖ You must pass a support **`FragmentManager`** to your adapter's base

```
public abstract class FragmentPagerAdapter : Android.Support.V4.View.PagerAdapter
{  ...
   public FragmentPagerAdapter(Android.Support.V4.App.FragmentManager fm)
   {
      ...
   }
}
```

Your adapter's constructor needs to chain to this base constructor and pass the manager

Adapter fragments

- Your adapter provides the Fragments to the **`ViewPager`**

```
public class MyAdapter : Android.Support.V4.App.FragmentPagerAdapter
{
   Android.Support.V4.App.Fragment[] fragments;

   public MyAdapter(Android.Support.V4.App.FragmentManager fm, Android.Support.V4.App.Fragment[] fragments)
      : base(fm)
   {
      this.fragments = fragments;
   }

   public override int Count
   {
      get { return fragments.Length; }
   }

   public override Android.Support.V4.App.Fragment GetItem(int position)
   {
      return fragments[position];
   }
}
```

Number of Fragments

Fragment at the given position

Xamarin University

Using an adapter

- You instantiate your adapter and load it into your **`ViewPager`**

```
protected override void OnCreate(Bundle bundle)
{
   var fragments = new Android.Support.V4.App.Fragment[]
   {
      new SessionFragment(),
      new SpeakerFragment(),
      new AboutFragment()
   };

   var viewPager = FindViewById<Android.Support.V4.View.ViewPager>(Resource.Id.viewPager);

   viewPager.Adapter = new MyAdapter(base.SupportFragmentManager, fragments);
}
```

2. Assign   1. Create

Individual Exercise

Implement gestural navigation with ViewPager

Summary

1. Add a **`ViewPager`** to your UI
2. Code an adapter to supply the **`ViewPager`** with Fragments

Combine gestural and tab navigation

Xamarin University

Tasks

1. Use a `ViewPager` to populate a `TabLayout` with tabs

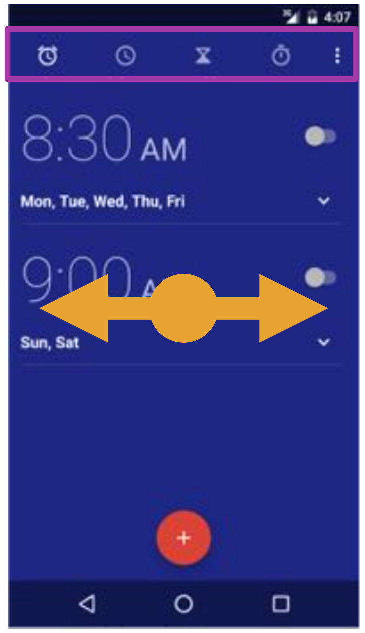
Xamarin University

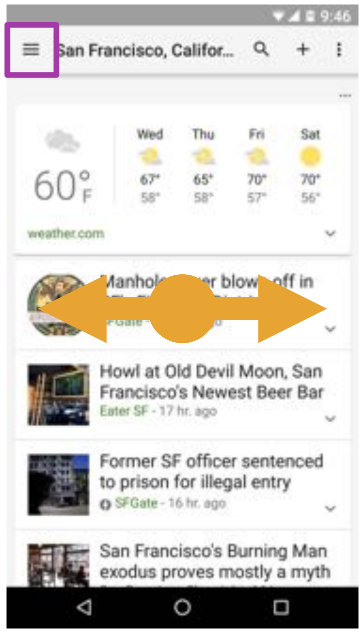
Hybrid navigation

❖ Many apps supplement their primary navigation with gestures

Can tap to navigate

Can swipe to navigate





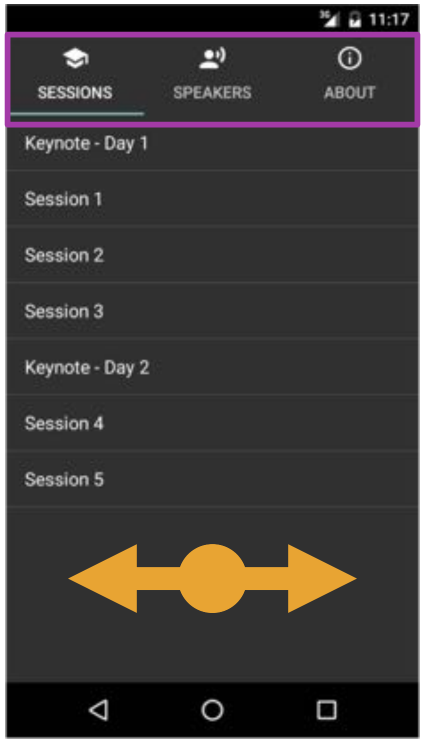
9:46
San Francisco, Califor...
60°F
Wed Thu Fri Sat
67° 65° 70° 70°
58° 58° 57° 56°
weather.com
Howl at Old Devil Moon, San Francisco's Newest Beer Bar
Eater SF - 17 hr. ago
Former SF officer sentenced to prison for illegal entry
SFGate - 16 hr. ago
San Francisco's Burning Man exodus proves mostly a myth

Xamarin University

TabLayout and ViewPager

❖ **`TabLayout`** and **`ViewPager`** know how to work together

`TabLayout` tells **`ViewPager`** when the user taps a tab so **`ViewPager`** can navigate to the selected page



Tab text is populated from **`ViewPager`** data (you have to set the icons manually via code)

`ViewPager` tells **`TabLayout`** when the user swipes so **`TabLayout`** can update the current tab

Xamarin University

TabLayout and ViewPager association

- **`TabLayout`** can be associated with a **`ViewPager`** and they will then automatically cooperate

```
public class TabLayout : HorizontalScrollView
{  ...
   public virtual void SetupWithViewPager(ViewPager viewPager)
   {
       ...
   }
}
```

The data from the **`ViewPager`** is used to create the tabs

Tabs + gesture [Steps]

- There are several steps needed to use a **`ViewPager`** with a **`TabLayout`**

1. Include Adapter titles
2. Create **`TabLayout`** and **`ViewPager`**
3. Create an Adapter
4. Associate **`TabLayout`** and **`ViewPager`**
5. (Optional) Set icons on the tabs

Tabs + gesture [Step 1]

❖ Your Adapter provides the tab titles to the **`ViewPager`**

Text type is not **`string`**

Tab text at the given position

```
public class MyAdapter : Android.Support.V4.App.FragmentPagerAdapter
{  ...
   ICharSequence[] titles;

   public MyAdapter(..., ICharSequence[] titles)
      : base(...)
   {
      this.titles = titles;
   }

   public override ICharSequence GetPageTitleFormatted(int position)
   {
      return titles[position];
   }
}
```

1 Include Adapter titles

Xamarin University

Tabs + gesture [Step 2]

- You need to create a **`TabLayout`** and a **`ViewPager`** (typically in XML)

`TabLayout` displays the tabs

`ViewPager` displays the pages

```
<LinearLayout ...>
  <android.support.design.widget.TabLayout
      android:id="@+id/tabLayout"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" />

  <android.support.v4.view.ViewPager
      android:id="@+id/viewPager"
      android:layout_width="match_parent"
      android:layout_height="wrap_content" />
</LinearLayout>
```

2 Create `TabLayout` and `ViewPager`

Tabs + gesture [Step 3]

Xamarin University

❖ Create an Adapter with Fragments and titles; load it into the **`ViewPager`**

```
var fragments = new Android.Support.V4.App.Fragment[]
{
   new SessionManager(),
   new SpeakerManager(),
   new AboutFragment()
};

var titles = CharSequence.ArrayFromStringArray(new[] { "Sessions", "Speakers", "About" });

var viewPager = FindViewById<ViewPager>(Resource.Id.viewPager);

viewPager.Adapter = new MyAdapter(base.SupportFragmentManager, fragments, titles);
```

Library method to convert strings to **`ICharSequence`**

Pages

Tab titles

3 Create an Adapter

Xamarin University

Tabs + gesture [Step 4]

- **TabLayout** has a method that takes a **ViewPager**, this associates the two and they will then automatically start working together

```
var viewPager = ...
...
var tabLayout = FindViewById<TabLayout>(Resource.Id.tabLayout);
tabLayout.SetupWithViewPager(viewPager);
```

Link the **TabLayout** to this **ViewPager**

4 Associate **TabLayout** and **ViewPager**

Xamarin University

Tabs + gesture [Step 5]

❖ You must set icons on the tabs manually; there is no support for automatic population via the **`ViewPager`**

```
var tabLayout = FindViewById<TabLayout>(Resource.Id.tabLayout);

tabLayout.SetupWithViewPager(viewPager);

tabLayout.GetTabAt(0).SetIcon(Resource.Drawable.sessions);
tabLayout.GetTabAt(1).SetIcon(Resource.Drawable.speakers);
tabLayout.GetTabAt(2).SetIcon(Resource.Drawable.about);
```

If you want your tabs to display icons, you must set them manually via code

5 (Optional) Set icons on the tabs

Individual Exercise

Combine gestural and tab navigation

Xamarin University

Summary

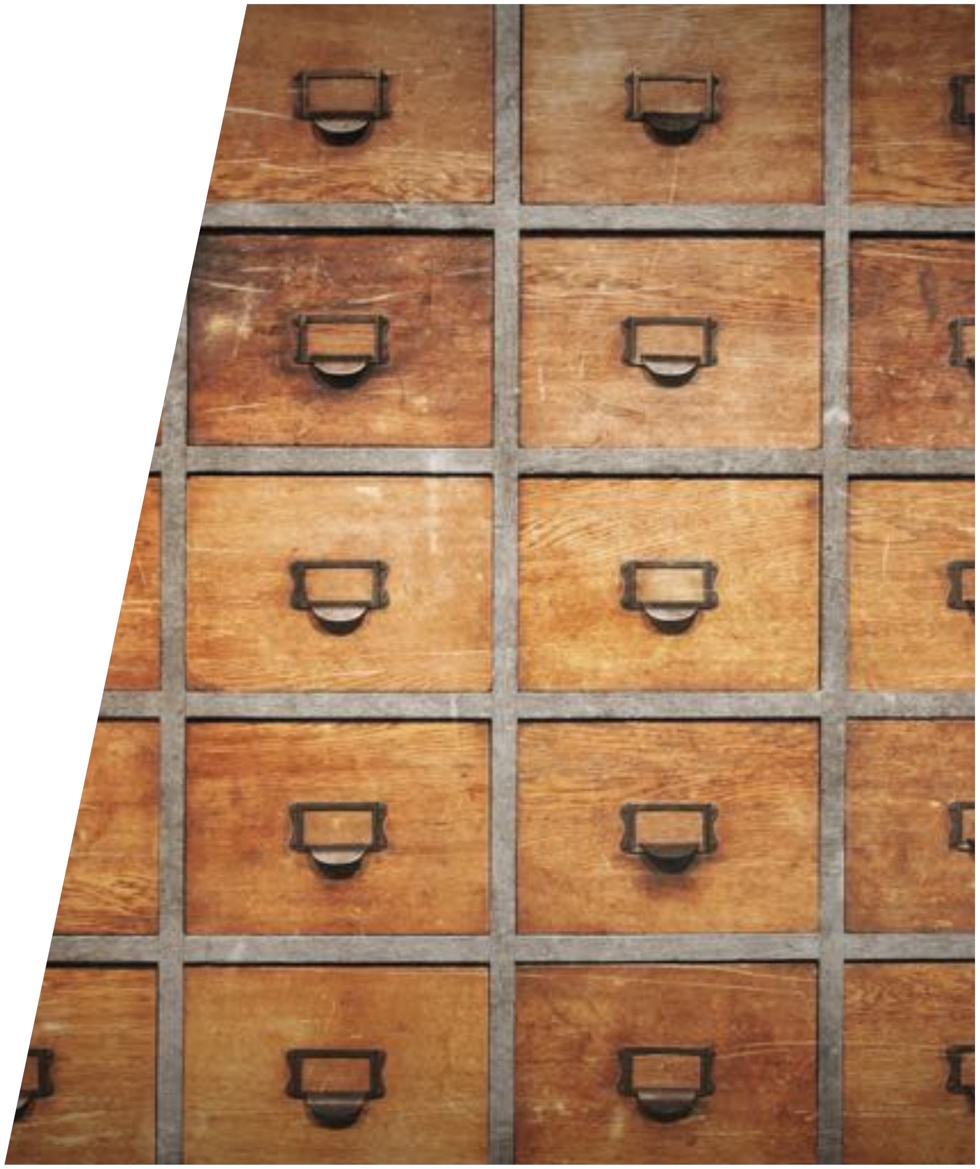
1. Use a `ViewPager` to populate a `TabLayout` with tabs

Implement drawer navigation with DrawerLayout

Xamarin University

Tasks

1. Use `DrawerLayout` to display a drawer menu
2. Use an app bar navigation button to open the drawer
3. Code an XML file for your menu
4. Swap content when your menu is clicked



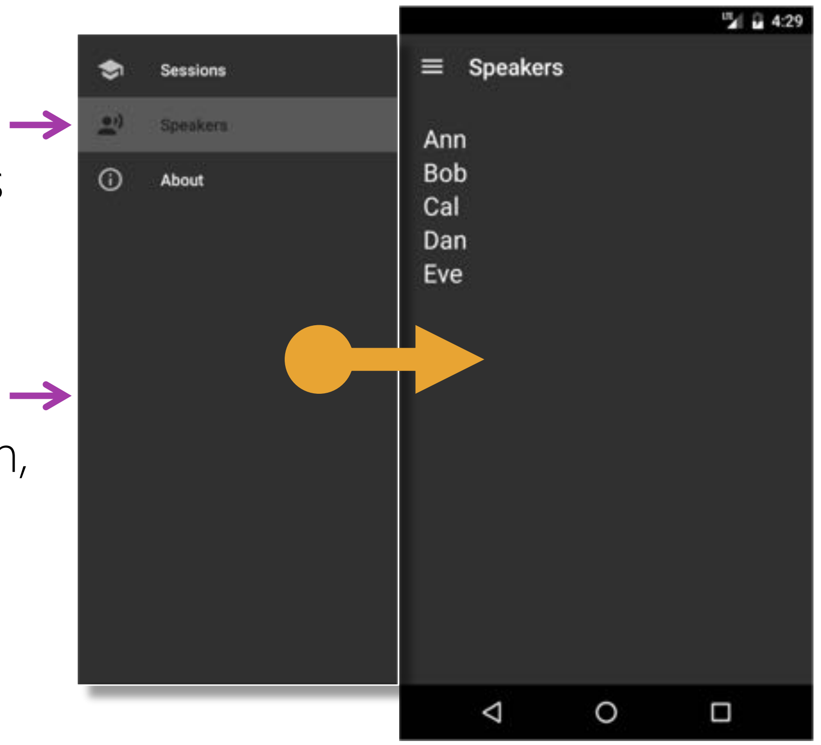
Xamarin University

What is drawer navigation?

- *Drawer navigation* uses a menu in a sliding panel for navigation

Menu has room for many entries

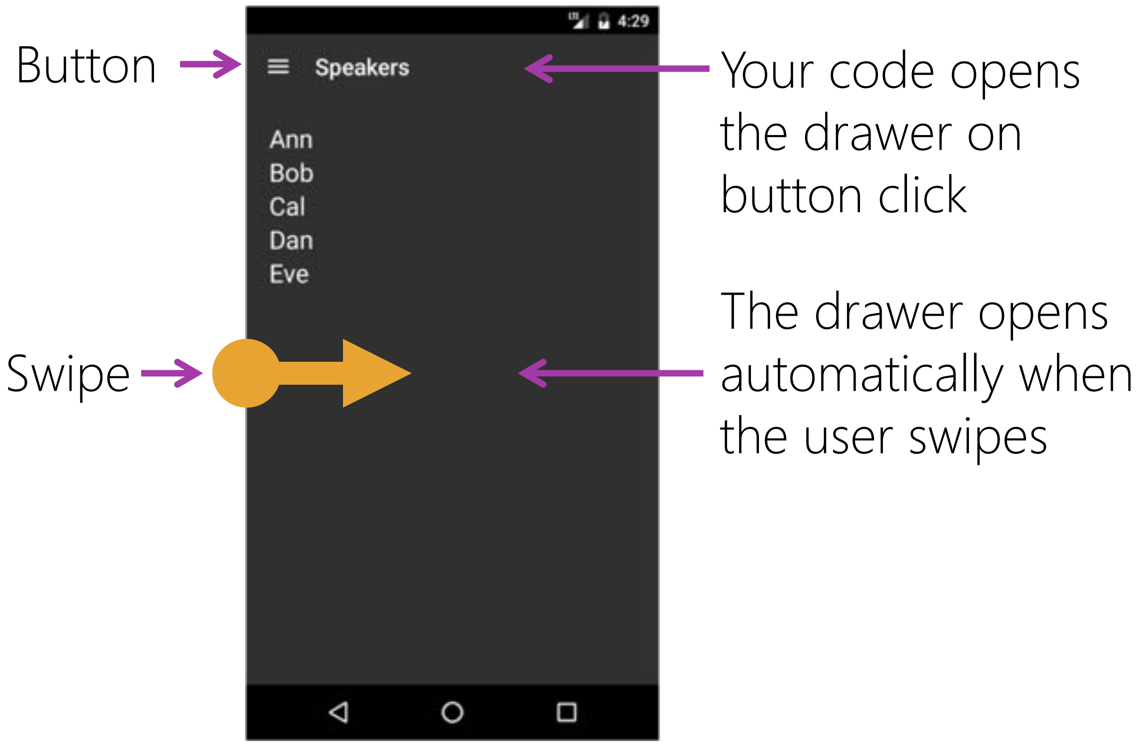
Menu overlays the content when open, but disappears when closed



Xamarin University

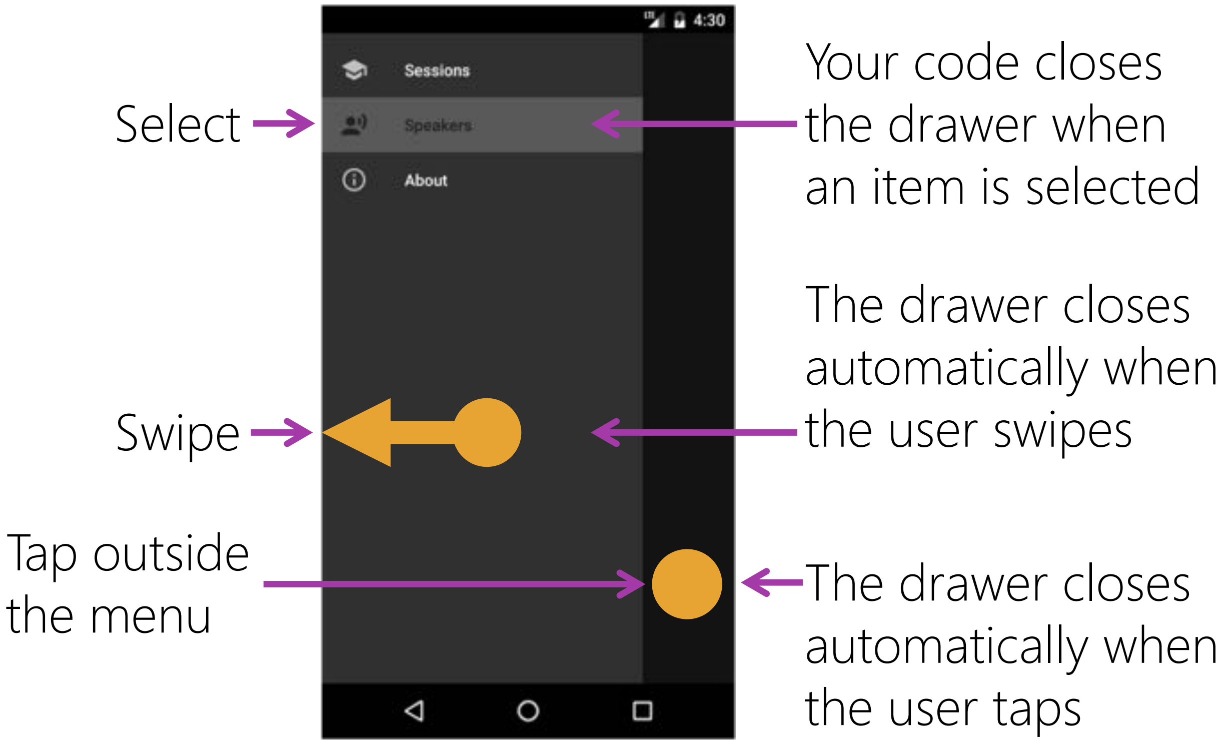
How to open the drawer?

- There are two ways for the user to open the drawer


Button
4:29
Speakers
Ann
Bob
Cal
Dan
Eve
Swipe
Your code opens the drawer on button click
The drawer opens automatically when the user swipes

How to close the drawer?

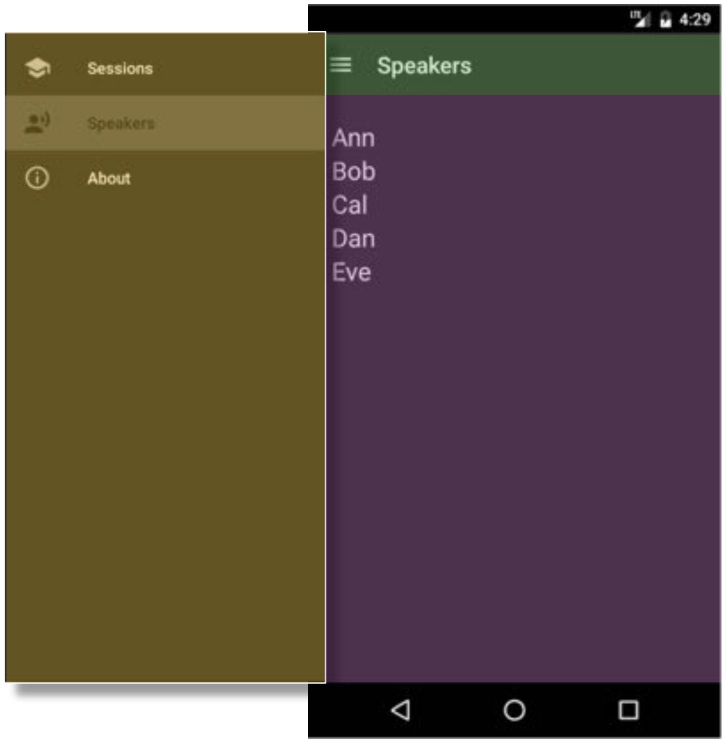
- There are three ways for the user to close the drawer


4:30
Sessions
Speakers
About
Select
Swipe
Tap outside the menu
Your code closes the drawer when an item is selected
The drawer closes automatically when the user swipes
The drawer closes automatically when the user taps

Layout structure

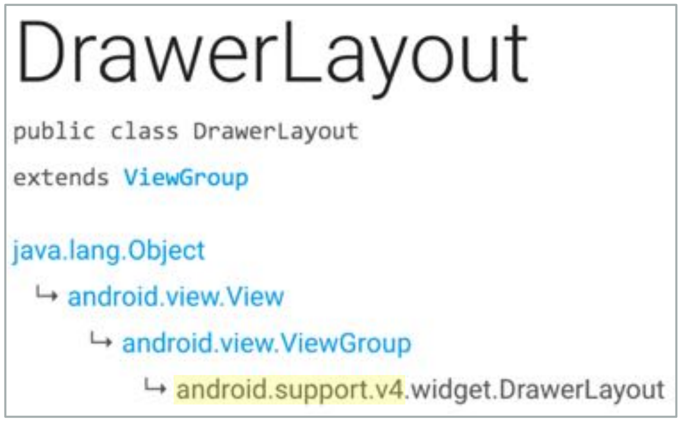
- It is typical to define the UI in XML with **`DrawerLayout`** as the root

```
<android.support.v4.widget.DrawerLayout ... >

   <LinearLayout ... >
      <android.support.v7.widget.Toolbar ... />
      <FrameLayout ... />
   </LinearLayout>

   <android.support.design.widget.NavigationView ... />

</android.support.v4.widget.DrawerLayout>
```



Required libraries

❖ The types that implement drawer navigation are in the support libraries

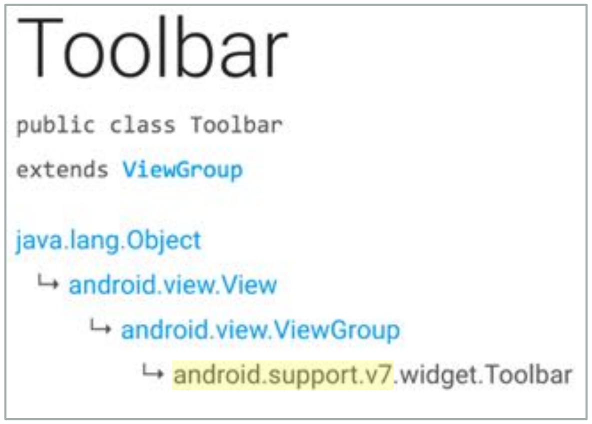


DrawerLayout

```
public class DrawerLayout
extends ViewGroup
```

java.lang.Object
  ↳ android.view.View
      ↳ android.view.ViewGroup
          ↳ android.support.v4.widget.DrawerLayout

v4 Support Library

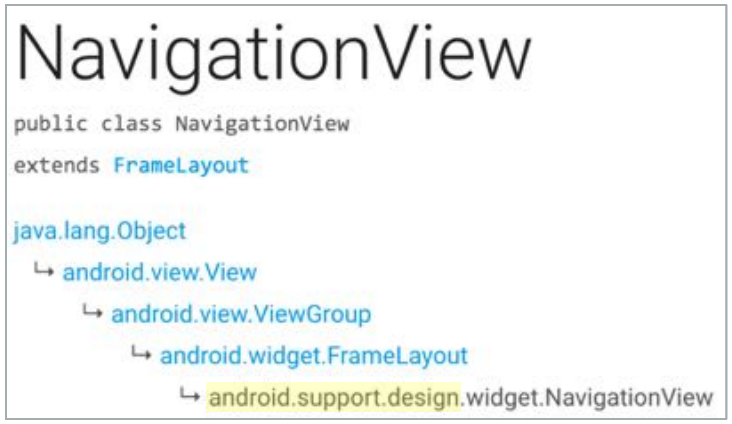


Toolbar

```
public class Toolbar
extends ViewGroup
```

java.lang.Object
  ↳ android.view.View
      ↳ android.view.ViewGroup
          ↳ android.support.v7.widget.Toolbar

v7 Support Library



NavigationView

```
public class NavigationView
extends FrameLayout
```

java.lang.Object
  ↳ android.view.View
      ↳ android.view.ViewGroup
          ↳ android.widget.FrameLayout
              ↳ android.support.design.widget.NavigationView

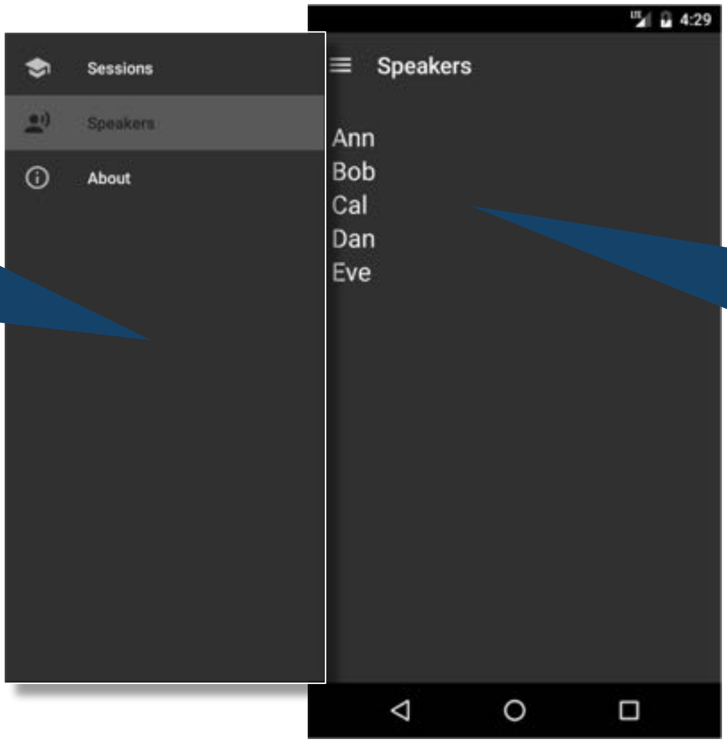
Design Support Library

We will use the support Toolbar as our Activity's app bar so our Activity will inherit from **`AppCompatActivity`** and we will use the Theme.AppCompat.NoActionBar app theme.

What is DrawerLayout?

❖ `DrawerLayout` is a layout manager that provides a flyout menu

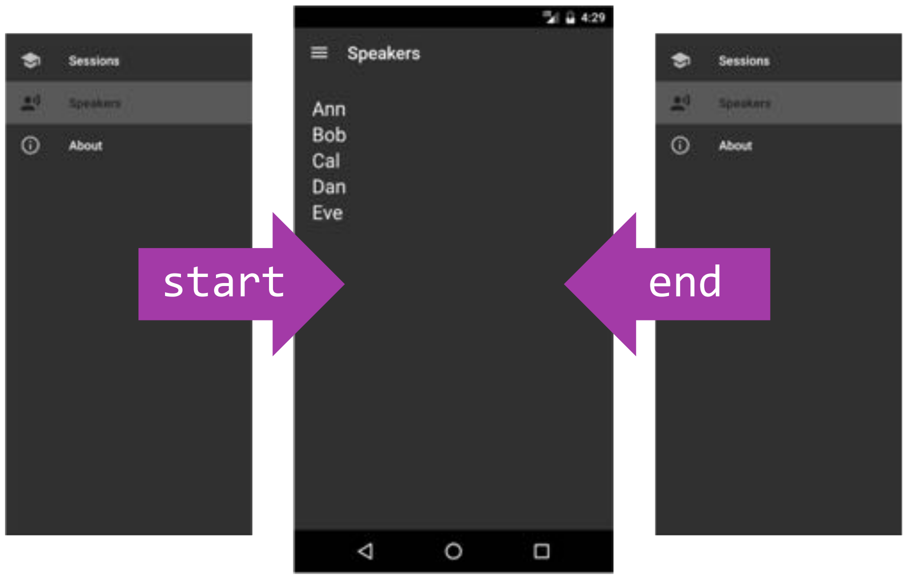
It implements the "sliding drawer" containing your menu



It is a layout manager that displays children for you

Drawer gravity

- `DrawerLayout` supports two drawers – each drawer is identified by its layout gravity



We will only cover the single-drawer case since it is the most common scenario

Xamarin University

DrawerLayout drawer management

- `DrawerLayout` lets you control the drawer, you specify the gravity to identify which drawer to open/close

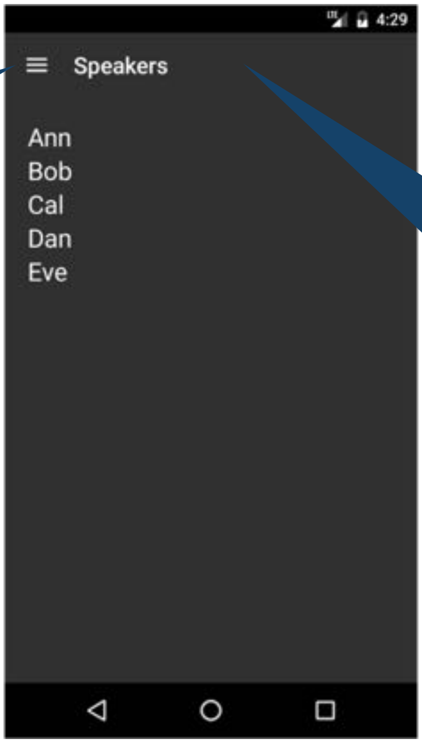
```
public class DrawerLayout : ViewGroup
{  ...
   public void OpenDrawer  (int gravity) { ... }
   public void CloseDrawer (int gravity) { ... }
}
```

You pass either `GravityCompat.Start` or `GravityCompat.End`

Xamarin University

Open button

❖ You should host your menu-open button in your Activity's app bar



Typical to use the Google **`ic_menu`** icon for your app bar's navigation icon

Typical to use a **`Toolbar`** as your Activity's app bar

Drawer open

- You open the drawer when the user clicks on the navigation button

```
public class MainActivity : AppCompatActivity
{
   DrawerLayout drawerLayout;

   protected override void OnCreate(Bundle savedInstanceState)
   {  ...
      drawerLayout = FindViewById<DrawerLayout>(Resource.Id.drawerLayout);
      ...
   }
   public override bool OnOptionsItemSelected(IMenuItem item)
   {
      switch (item.ItemId)
      {
         case Android.Resource.Id.Home: drawerLayout.OpenDrawer(GravityCompat.Start); break;
      }
      return true;
   }
}
```

App bar nav button clicks reported here

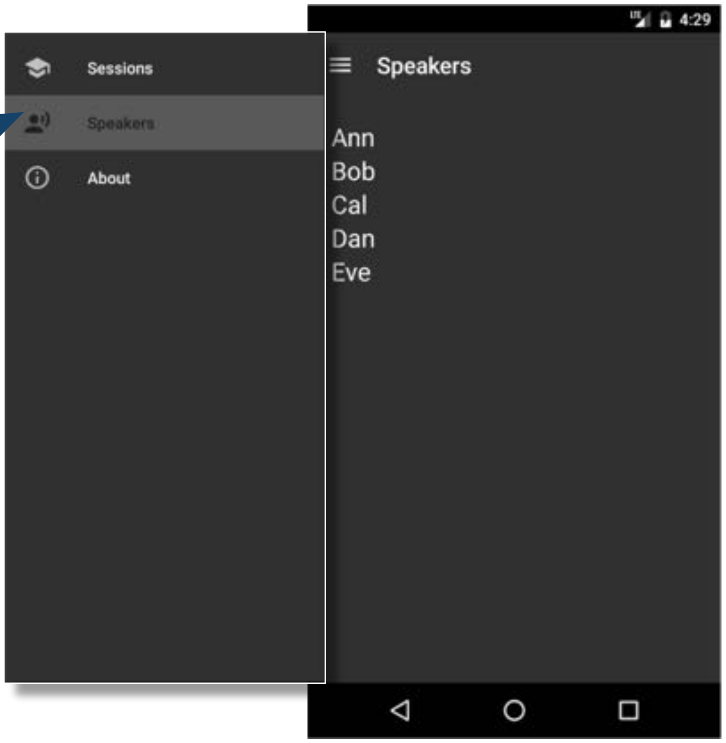
App bar nav button uses this standard Android ID

Xamarin University

What is NavigationView?

❖ `NavigationView` implements a navigation menu

It manages the menu hosted inside the flyout



DrawerLayout/NavigationView association

❖ **`DrawerLayout`** searches its children for a **`NavigationView`** and automatically uses it for the flyout menu when found

```
<android.support.v4.widget.DrawerLayout ... >

   <LinearLayout ... >
      <android.support.v7.widget.Toolbar ... />
      <FrameLayout ... />
   </LinearLayout>

   <android.support.design.widget.NavigationView ... />

</android.support.v4.widget.DrawerLayout>
```

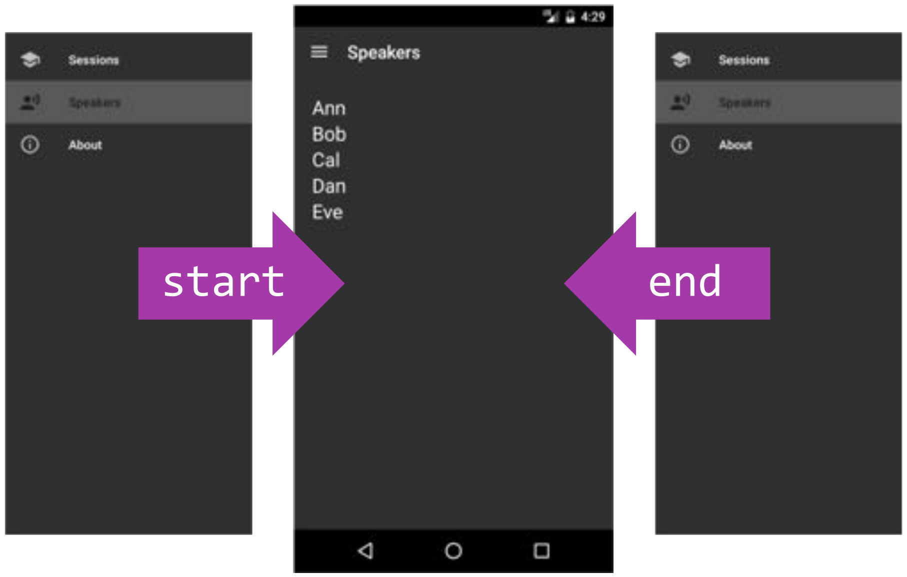
`NavigationView` will automatically become the drawer's menu

Xamarin University

Drawer gravity

- Set layout gravity on the **`NavigationView`** to specify the side the drawer enters (failure to set it yields a runtime exception)

```
<android.support.design.widget
  .NavigationView
  android:layout_gravity="start"
  ... />
```



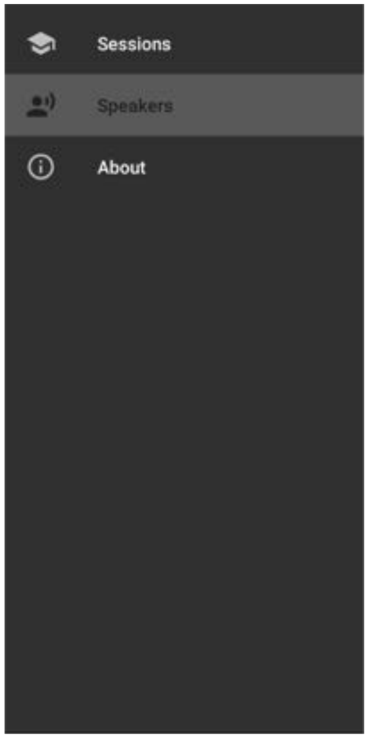
Xamarin University

NavigationView menu definition

❖ `NavigationView`'s menu is created from an Android XML menu file

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/sessionsMenuItem"
            android:icon="@drawable/ic_school_white_24dp"
            android:title="Sessions" />
        <item
            android:id="@+id/speakersMenuItem"
            android:icon="@drawable/ic_record_voice_over_white_24dp"
            android:title="Speakers"/>
        <item
            android:id="@+id/aboutMenuItem"
            android:icon="@drawable/ic_info_outline_white_24dp"
            android:title="About"/>
    </group>
</menu>
```

One item selected at a time



NavigationView menu loading

Xamarin University

- Set the **`NavigationView`**'s **`menu`** property to the name of the menu file

`Resources/menu/navigation_menu.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<menu ...>
   ...
</menu>
```

Place the file in the menu folder

```
<android.support.design.widget.NavigationView
   xmlns:app="http://schemas.android.com/apk/res-auto"
   app:menu="@menu/navigation_menu"
   ...
   />
```

Use the res-auto namespace prefix

Specify the file in the declaration for your `NavigationView`

Xamarin University

NavigationView item selection

❖ **NavigationView**'s event notifies you when the user selects an item

```
public class MainActivity : Android.Support.V7.App.AppCompatActivity
{ ...
   protected override void OnCreate(Bundle savedInstanceState)
   {   ...
      var menu = FindViewById<NavigationView>(Resource.Id.navigationView);
      ...
      menu.NavigationItemSelected += OnSelected;
   }

   void OnSelected(object sender, NavigationView.NavigationItemSelectedEventArgs e)
   {  ...
      switch (e.MenuItem.ItemId) { ...  }
   }
}
```

Event args give you the selected menu item

NavigationView navigation

❖ You navigate your app when the user selects a **`NavigationView`** item

```
void MenuItemSelected(object sender, NavigationView.NavigationItemSelectedEventArgs e)
{
   switch (e.MenuItem.ItemId)
   {
      case Resource.Id.sessionsMenuItem: Navigate(new SessionsFragment()); break;
      case Resource.Id.speakersMenuItem: Navigate(new SpeakersFragment()); break;
      case Resource.Id.aboutMenuItem:    Navigate(new AboutFragment   ()); break;
   }

   e.MenuItem.SetChecked(true);
   drawerLayout.CloseDrawer(Android.Support.V4.View.GravityCompat.Start);
}

void Navigate(Fragment fragment)
{
   var transaction = base.SupportFragmentManager.BeginTransaction();
   transaction.Replace(Resource.Id.contentFrame, fragment);
   transaction.Commit();
}
```

Determine which item was selected

Show the menu item as checked

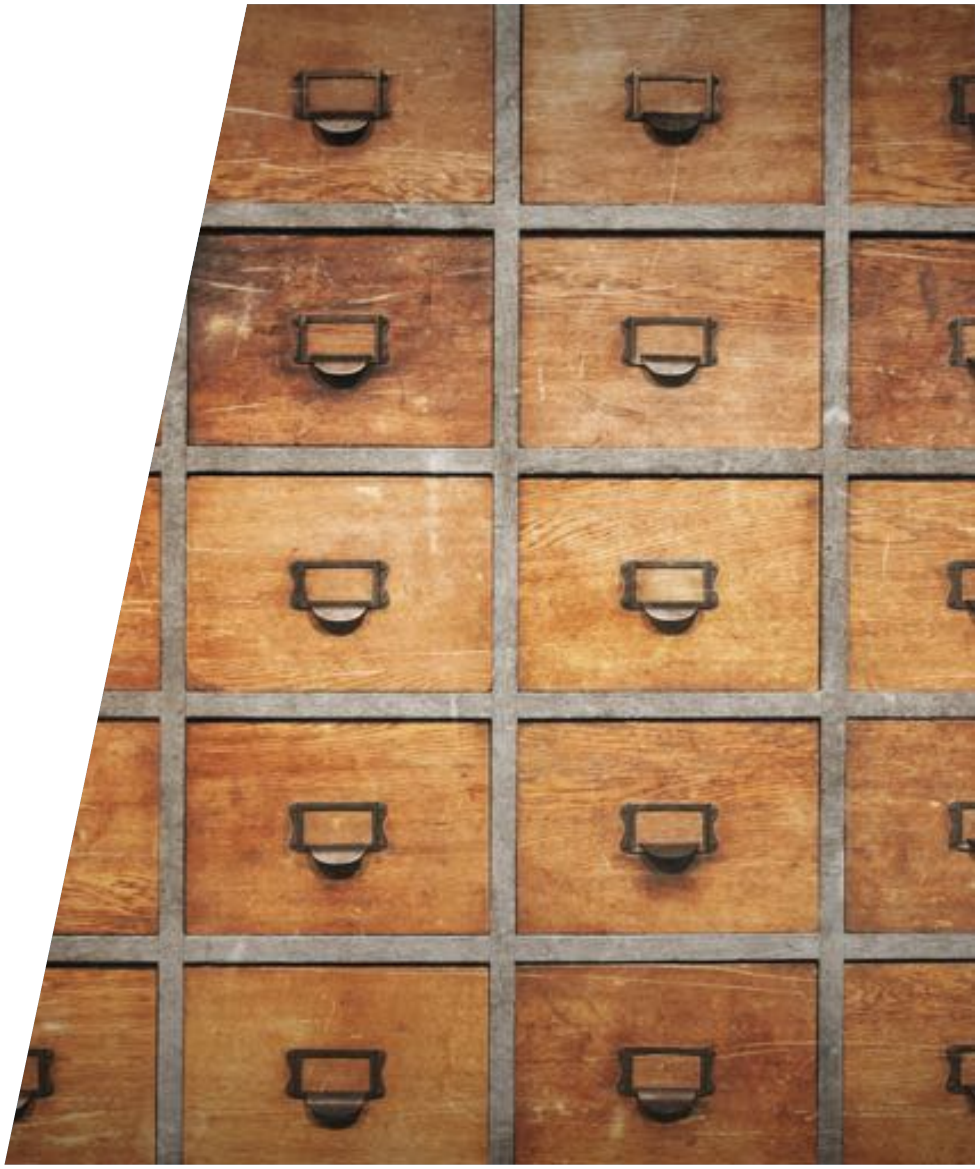
Close the drawer

Individual Exercise

Implement drawer navigation with DrawerLayout

Summary

1. Use `DrawerLayout` to display a drawer menu
2. Use an app bar navigation button to open the drawer
3. Code an XML file for your menu
4. Swap content when your menu is clicked



Thank You!

Please complete the class survey in your profile:
university.xamarin.com/profile

Microsoft