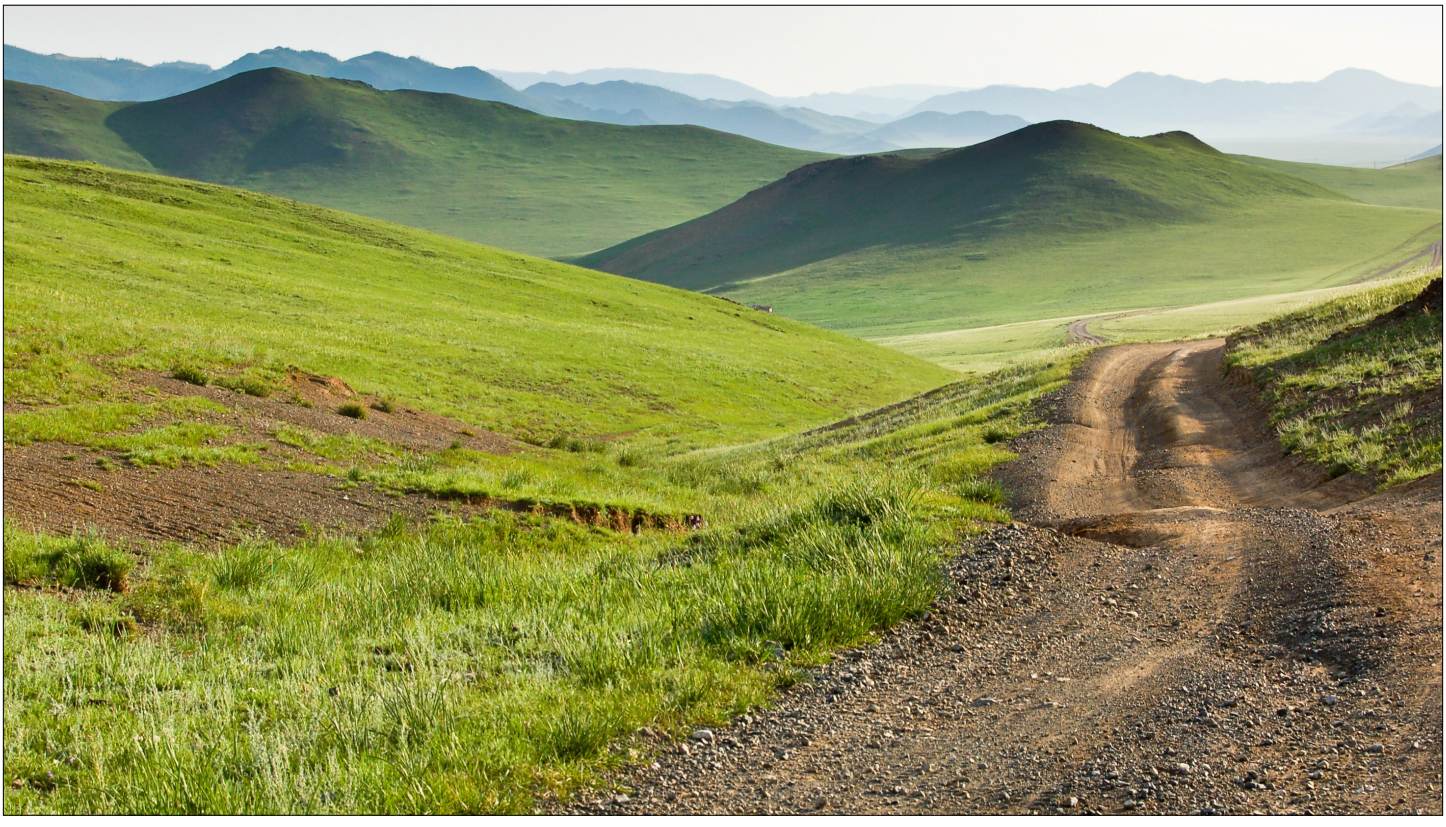
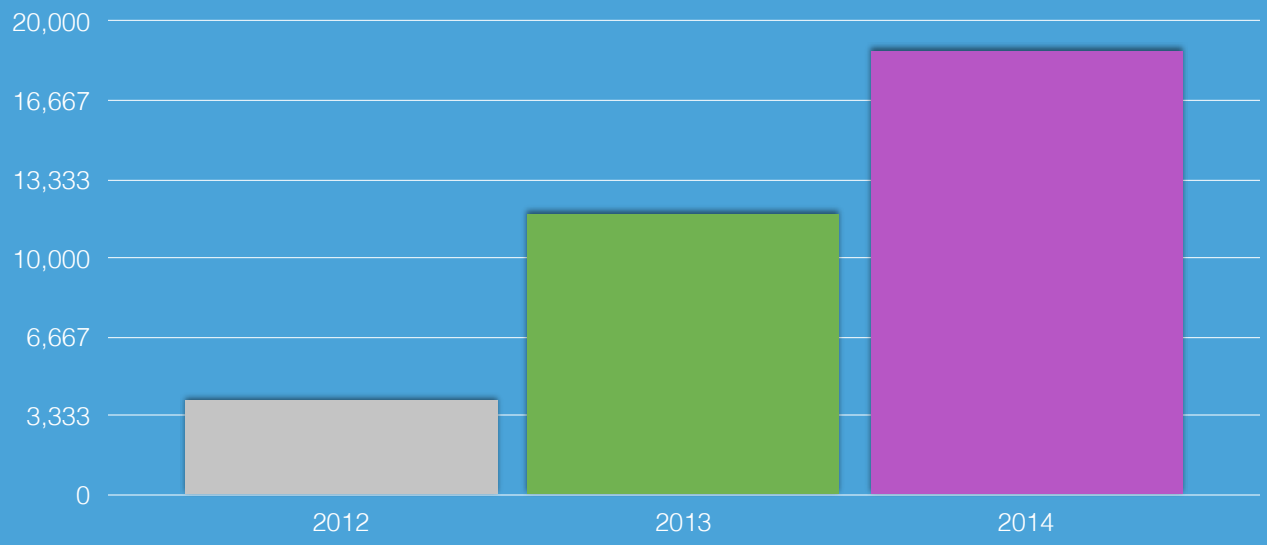
Xamarin Evolve 2014

Designing Android UIs for the Ever Changing Device Landscape

Michael Stonis
Eight-Bot, Inc.
@michaelstonis



Unique Android Devices



Open Signal, August 2014

Big Opportunity

Where do we get started?

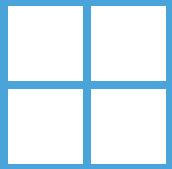
- Design Fundamentals
- Layout Management
- Android Fragments

Design Fundamentals

Density Independent Pixels

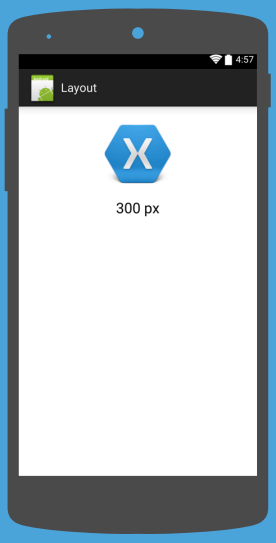
Xamarin Evolve 2014

Density Independent Pixel (DIP)

DPI	Pixels Per DIP
160	
320	

- Represents a single pixel on a 160 dpi screen
- Scales on screens with higher or lower density
- Use as a standard unit of measurement
- *Avoid Pixels Like the Plague!*

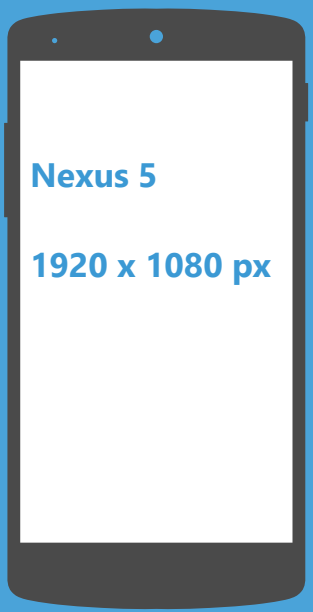
Xamarin Evolve 2014



Pixels

Density Independent Pixels

Density Independent Pixels



Density Independent Pixels

1 inch

445 pixels

Xamarin Evolve 2014

Warning

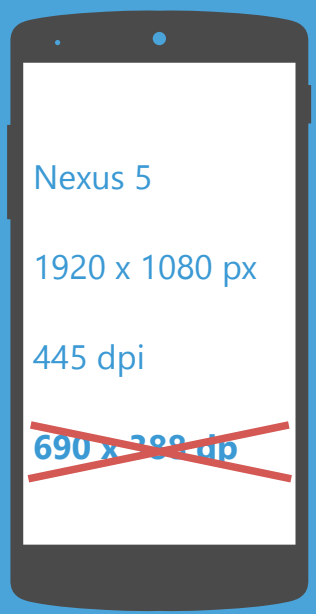
Math Ahead…Approach with Caution

Xamarin Evolve 2014

Calculating DIPs

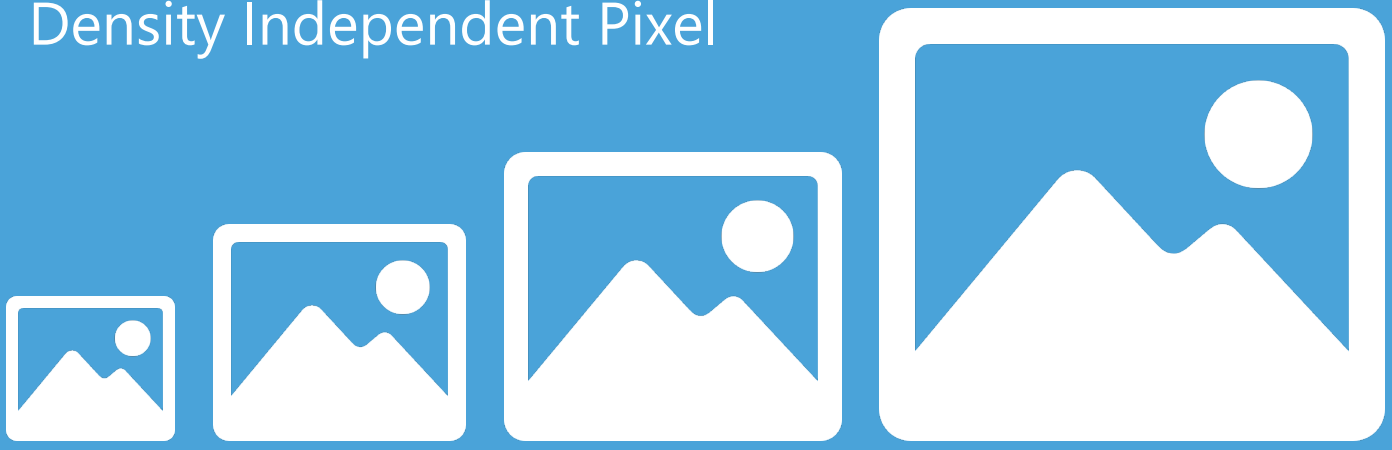
$$\textbf{DIP} = \frac{\text{Resolution (px) x } \mathbf{160}}{\text{dpi}}$$

Density Independent Pixels



$$\frac{\text{1920 px X 160}}{\text{445 dpi}}$$

Density Independent Pixel



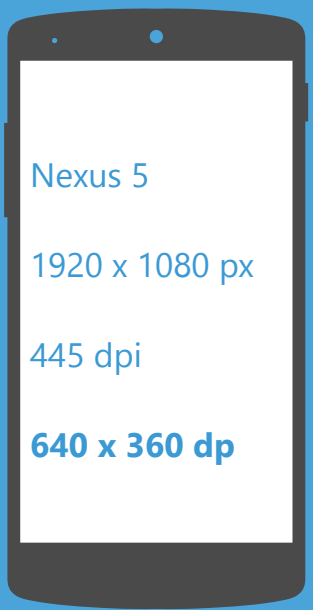
160 dpi
MDPI
1x

240 dpi
HDPI
1.5x

320 dpi
XHDPI
2x

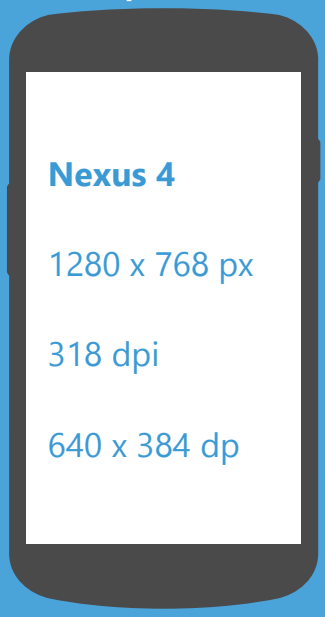
480 dpi
XXHDPI
3x

Density Independent Pixels



1920 px X 160
480 dpi

Device Comparison

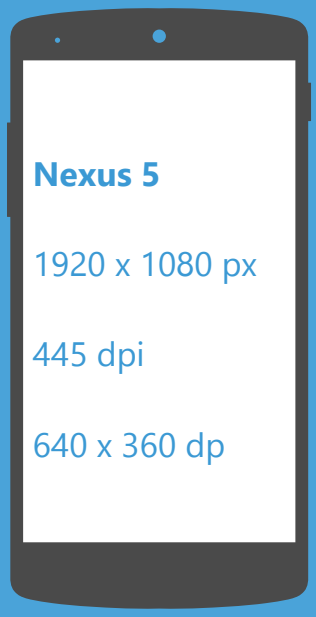


Nexus 4

1280 x 768 px

318 dpi

640 x 384 dp



Nexus 5

1920 x 1080 px

445 dpi

640 x 360 dp

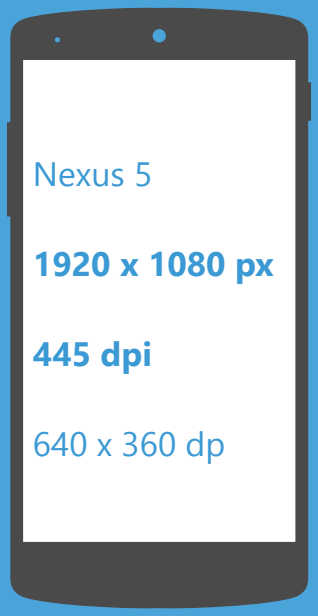
Device Comparison

Nexus 4

1280 x 768 px

318 dpi

640 x 384 dp



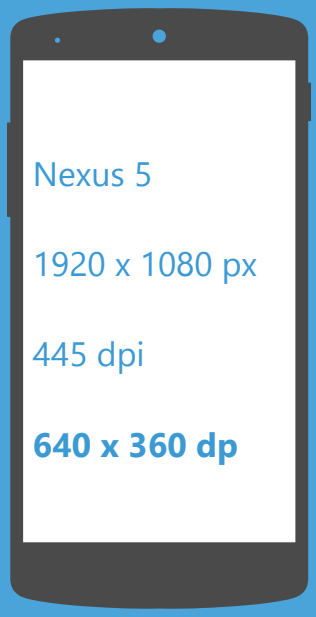
Nexus 5

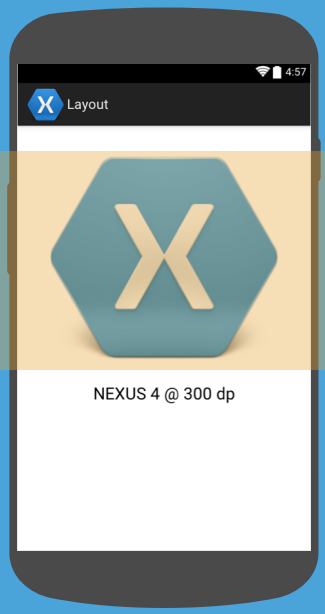
1920 x 1080 px

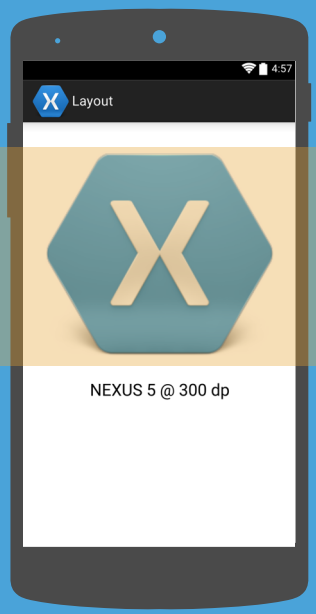
445 dpi

640 x 360 dp

Device Comparison







Android Resource Qualifiers

Xamarin Evolve 2014

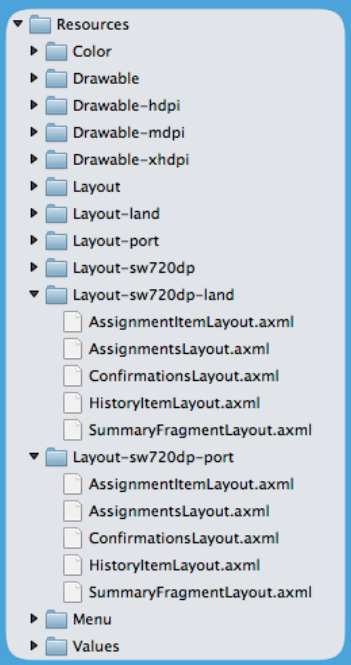
Android Resource Qualifiers

Different Strokes for Different Folks

- Provides us the ability to use different resources for different configurations

- Qualifiers Apply to Resource Folders
  - drawable
  - layout
  - etc.

- *Do not apply to resources directly*



Xamarin Evolve 2014

Android Resource Qualifiers

Qualifier Format

<resource type>-<qualifier>

Android Resource Qualifiers

Example

drawable-large-xhdpi

A drawable resource on a large screen size with an Extra-high-density screen

Android Resource Qualifiers

Example

layout-zh-land

A layout resource for the Chinese language presented in landscape orientation

Android Resource Qualifiers

Resource Qualifier Types

- Region & Language
  - Mobile Country Code (MCC)
  - Language and Language Region
- Device Configuration
  - Portrait
  - Landscape
- Device Size Resolution
  - Screen Size
  - Pixel Density
- Device Features
  - UI Mode
  - Hardware Software Keyboard

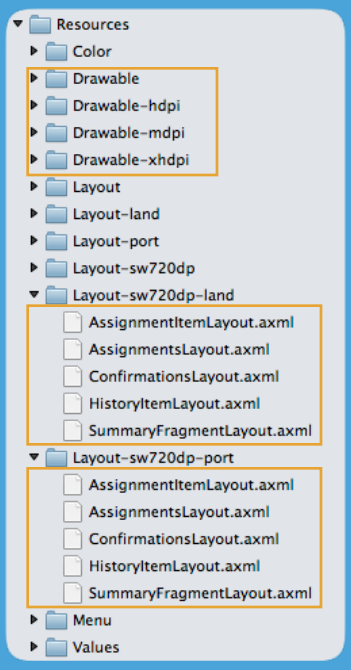
Android Resource Qualifiers

Different Strokes for Different Folks

- Provides us the ability to use different resources for different configurations

- Qualifiers Apply to Resource Folders
  - drawable
  - layout
  - etc.

- *Do not apply to resources directly*

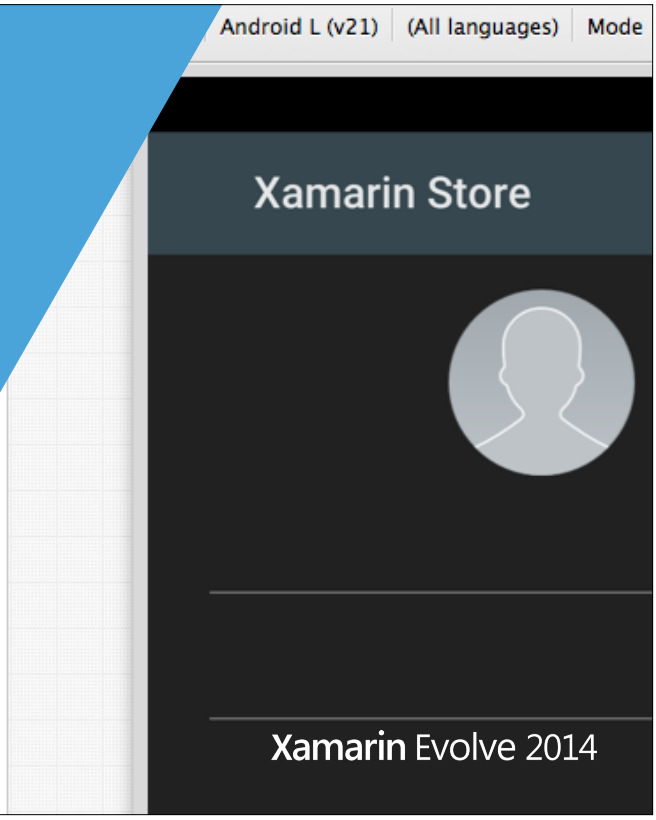


Layout Management

Layout Management

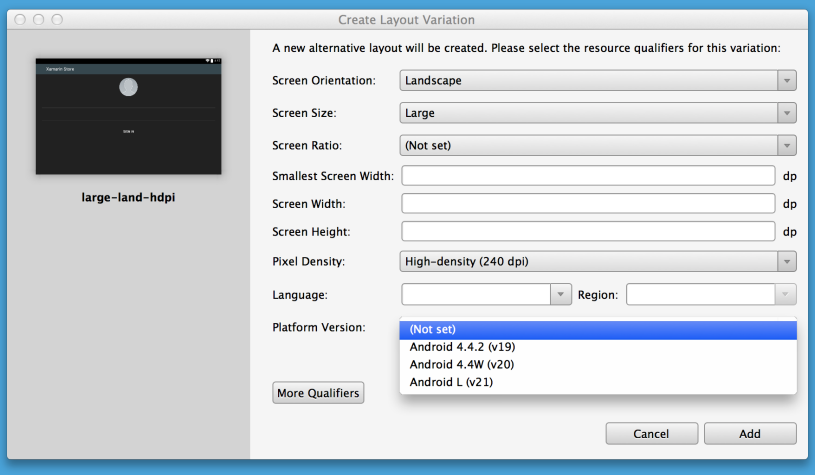
Xamarin Android Designer

- Full WYSIWYG Editor for Android
  - Control Management
  - Document Explorer
  - XML Editor
- Layouts are standard Android
- Supports Editing *Multiple* Layouts



Layout Management

Multiple Layouts



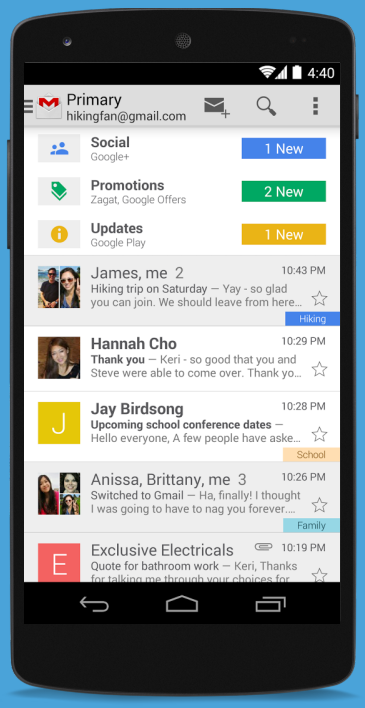
Demo

Layout Management

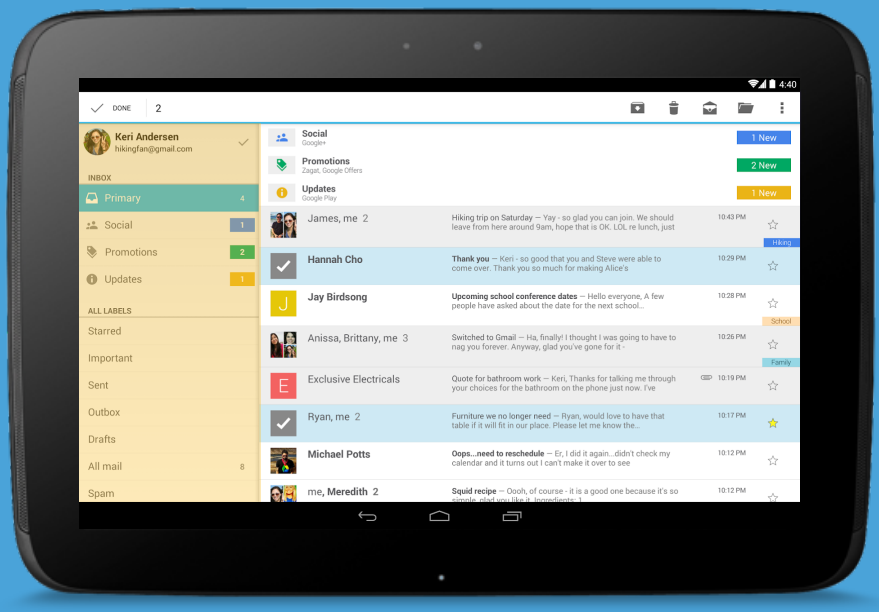
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  android:versionCode="1" android:versionName="1.0"
  package="com.xamarin.university.mobilenav.android">
  <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="14" />
  <application android:icon="@drawable/ic_launcher" android:label="ActionBar Tabs">
  </application>
  <supports-screens android:compatibleWidthLimitDp="integer"/>
</manifest>
```

Fragments

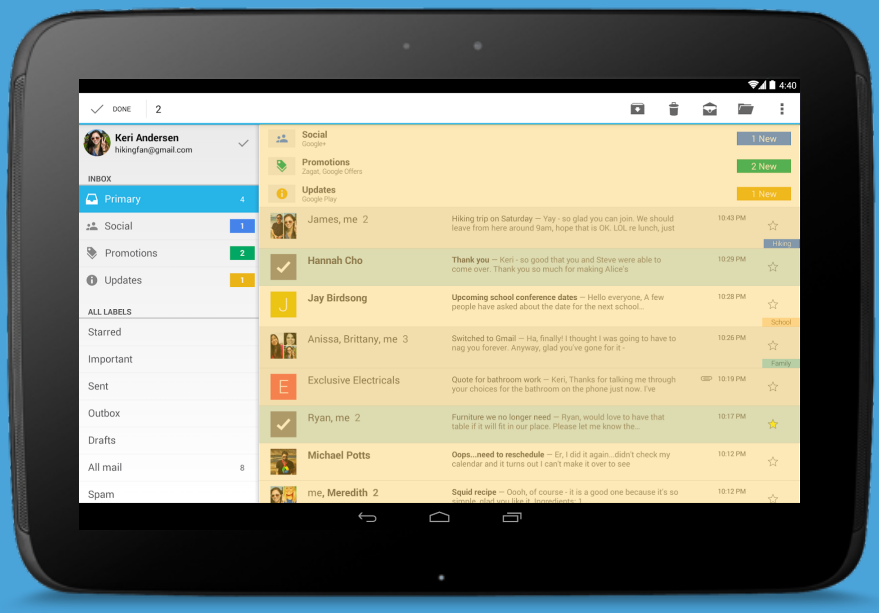
Xamarin Evolve 2014



Xamarin Evolve 2014



Xamarin Evolve 2014



Xamarin Evolve 2014

Android Fragments

Application Building Blocks

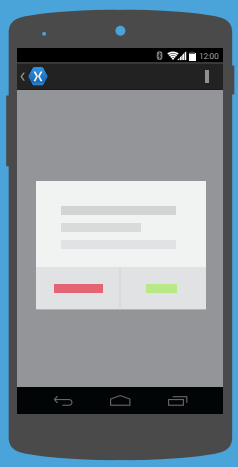
- Reusable User-Interface components
- Conceptually similar to an Activity with its own lifecycle and layout
- Create in XML or programmatically

Android Fragment Creation

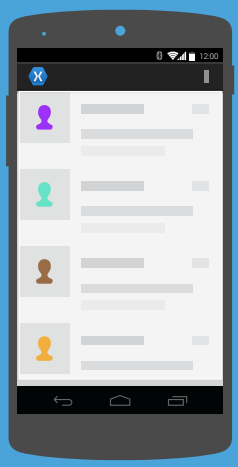
```
public class FragmentToAdd : Fragment
{
  public override void OnCreate (Bundle savedInstanceState)
  {
  }

  public override View OnCreateView (LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState)
  {
  }

  public override void OnPause ()
  {
  }
}
```

Android Fragment Key Lifecycle Methods

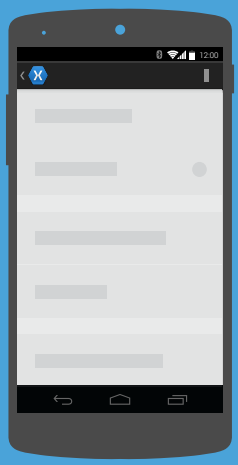
```
public class FragmentToAdd : Fragment
{
  public override void OnCreate (Bundle savedInstanceState)
  {
  }

  public override View OnCreateView (LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState)
  {
  }

  public override void OnPause ()
  {
  }
}
```

Dialog



List

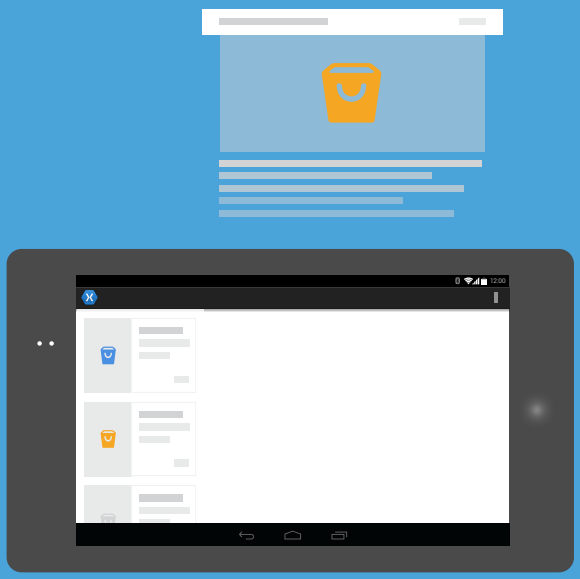


Preferences

Xamarin Evolve 2014

Adding a Fragment

Xamarin Evolve 2014



Xamarin Evolve 2014

Adding Android Fragments via XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:minWidth="25px"
    android:minHeight="25px">
    <fragment
        class="fragments.NavigationFragment"
        android:id="@+id/navigation"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" />
</LinearLayout>
```


Adding Android Fragments via Code

```
using (var transaction = FragmentManager.BeginTransaction())
{
    transaction.Add(Resource.Id.container, new FragmentToAdd());

    transaction.Commit();
}
```

Adding Android Fragments via Code

```
using (var transaction = FragmentManager.BeginTransaction())
{
    transaction.Add(Resource.Id.container, new FragmentToAdd());

    transaction.Commit();
}
```


Adding Android Fragments via Code

```
using (var transaction = FragmentManager.BeginTransaction())
{
    transaction.Add(Resource.Id.container, new FragmentToAdd());

    transaction.Commit();
}
```

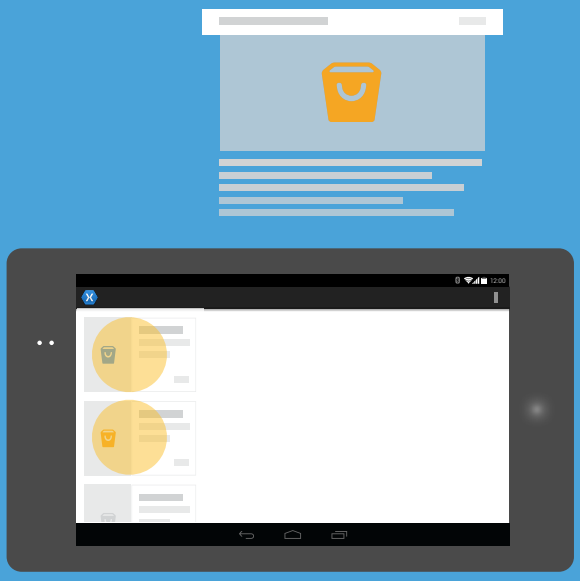
Xamarin Evolve 2014

Adding Android Fragments via Code

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <FrameLayout
        android:minWidth="25px"
        android:minHeight="25px"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:id="@+id/container"
        android:layout_weight="1" />
```


Adding Android Fragments via Code

```
using (var transaction = FragmentManager.BeginTransaction())
{
    transaction.Add(Resource.Id.container, new FragmentToAdd());

    transaction.Commit();
}
```



Replacing Android Fragments

```
using (var transaction = FragmentManager.BeginTransaction())
{
    transaction.SetCustomAnimations(
        Android.Resource.Animator.FadeIn,
        Android.Resource.Animator.FadeOut);

    transaction.Replace(Resource.Id.container, new FragmentToAdd());

    transaction.Commit();
}
```

Xamarin Evolve 2014

Communicating with Fragments

Xamarin Evolve 2014

Build for Modularity

Think like Building Blocks

- Assume that the Activity knows nothing about your Fragment and vice-versa
- Build knowing that it will be reused and used in varying different contexts
- Make it easy for your Fragment to discover host Activity Functionality

Xamarin Evolve 2014

Providing Data to a Fragment

```
var fragmentToAdd = new FragmentToAdd {
  Arguments = new Bundle ()
};

fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorRed, backgroundColor.R);
fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorGreen, backgroundColor.G);
fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorBlue, backgroundColor.B);
```


Guaranteeing Fragment Receives Data

```
public static FragmentToAdd BuildFragmentToAdd (Color backgroundColor)
{
  var fragmentToAdd = new FragmentToAdd {
    Arguments = new Bundle ()
  } ;

  fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorRed, backgroundColor.R);
  fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorGreen, backgroundColor.G);
  fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorBlue, backgroundColor.B);

  return fragmentToAdd;
}
```

Guaranteeing Fragment Receives Data

```
public static FragmentToAdd BuildFragmentToAdd (Color backgroundColor)
{
  var fragmentToAdd = new FragmentToAdd {
    Arguments = new Bundle ()
  } ;

  fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorRed, backgroundColor.R);
  fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorGreen, backgroundColor.G);
  fragmentToAdd.Arguments.PutInt (ArgumentBackgroundColorBlue, backgroundColor.B);

  return fragmentToAdd;
}
```

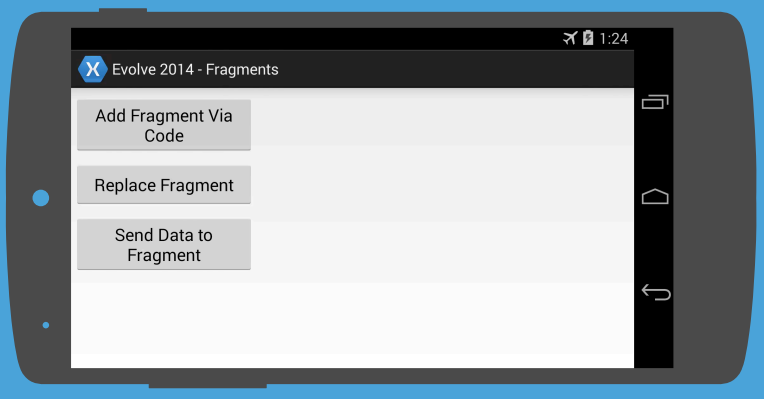

Messaging with the Host Activity

Passing notes in class is okay

- Create an Interface that defines any external functionality your fragment needs
- Implement the Interface in your Activity
- In the fragment, check if the Activity has implemented the Interface and make use of any defined functionality

Demo

Xamarin Evolve 2014



Xamarin Evolve 2014

Communicating with the Host Activity

```
public interface IDisplayDetail
{
  bool CanDisplayDetail { get; }

  void DisplayDetail(Fragment fragmentToDisplay);
}
```


Communicating with the Host Activity

```
var displayDetailActivity = Activity as IDisplayDetail;

if (displayDetailActivity != null && displayDetailActivity.CanDisplayDetail )
  displayDetailActivity.DisplayDetail (new FragmentToAdd ());
else
  StartActivity(new Intent(this.Activity, typeof(AddFragmentActivity)));
```

Communicating with the Host Activity

```
var displayDetailActivity = Activity as IDisplayDetail;

if (displayDetailActivity != null && displayDetailActivity.CanDisplayDetail)
  displayDetailActivity.DisplayDetail (new FragmentToAdd ());
else
  StartActivity(new Intent(this.Activity, typeof(AddFragmentActivity)));
```


Communicating with the Host Activity

```
var displayDetailActivity = Activity as IDisplayDetail;

if (displayDetailActivity != null && displayDetailActivity.CanDisplayDetail )
  displayDetailActivity.DisplayDetail (new FragmentToAdd ());
else
  StartActivity(new Intent(this.Activity, typeof(AddFragmentActivity)));
```

Xamarin Evolve 2014

Tying It All Together

Work together in harmony

- Fragments are individual modules, but they can work together
- Use an Activity to orchestrate communications between multiple fragments
- Mix-and-Match Fragments when space allows, such as when in landscape mode or on a tablet

One Last Thing…

Testing. Testing Never Changes

- Device manufacturers implement Android Differently
  - Camera
  - Storage
  - Event ListViews!
- Emulators Lie

Questions?

Xamarin Evolve 2014

Xamarin Evolve 2014

Designing Android UIs for the Ever Changing Device Landscape

Michael Stonis
@michaelstonis