UNIVERSITY of WASHINGTON

Android 210 - Lecture 4
Fragments & ActionBar

Margaret Maynard-Reid
February 2, 2014

Agenda

- ListView
- Intro to Fragment
- ActionBar & Menu
- ViewPager

Android Stories

- Android Distribution Updated for February 2015 – Lollipop Makes First Appearance at 1.6%
- Android shipments in 2014 exceed 1 billion for first time
- Android Wear & QR Code: Putting Users through the Fast Track

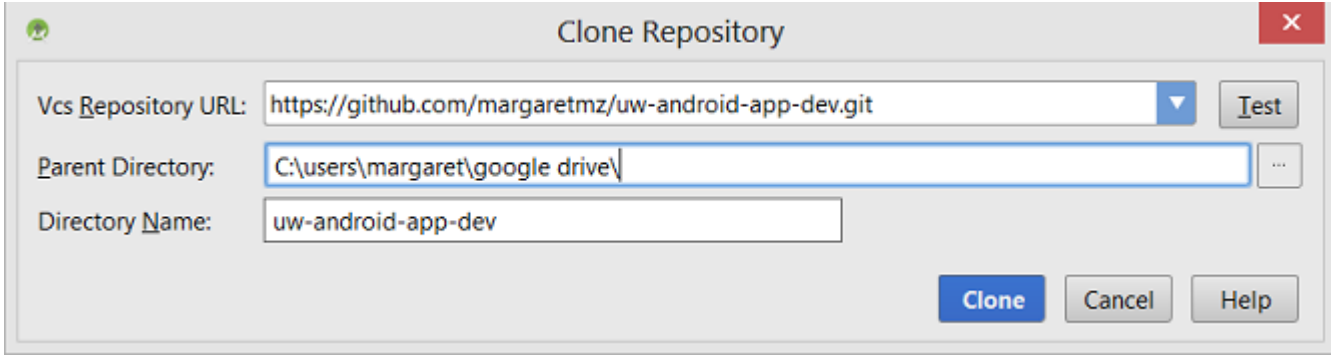
Final Project

- Great work on the project idea/scope!
- Finalize UI design by end of first course
- Focus on just a few features
- Showcase what you've learned

Sample Code Posted on GitHub

In Android Studio, Git/Checkout from Version Control/Git

https://github.com/margaretmz/uw-android-app-dev.git



Homework 1

- No need to nest the TextViews in LinearLayout in order to position them
- Try use Framelayout if there is only one child
- No need for a separate layout for landscape for every screen
- It’s OK if you didn’t have the most efficient layout.

Review of last week

- What is an EditText?
- How to you restore EditText state?
- Can you interact with a Toast?
- Main components in creating an AlertDialog?
- What is a Spinner?
- What is a ListView?
- What is an Adapter?

ListView

- A view group that displays items in a vertically scrolling list.
- The items come from the adapter associated with the ListView.
- For now we will only use an ArrayAdapter

ArrayAdapter

- Backed by an array of arbitrary objects.
- By default expects that the provided resource id references **a single TextView**.
- By default invokes toString() on the objects and display them in the TextView widgets.
- For custom list item layout, use the constructors that also takes a field id. That field id should reference a TextView in the larger layout resource.

http://developer.android.com/reference/android/widget/ArrayAdapter.html

How to create a simple list?

1. Include ListView in an xml layout
2. Find the ListView by Id in activity

   ListView listView = (ListView)findViewById(R.id.listView)
3. Create an ArrayAdapter

   ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
   android.R.layout.simple_list_item_1, androidAPIs);
4. listview.SetAdapter(adapter)

ArrayAdatper - for simple list

ArrayAdapter constructor parameters:

1. Context ← activity instance
2. Resource ID of a view for the list item ← use Android layout or your own. Root must be a single TextView!
3. The array or list of data

Custom ArrayAdapter

- Define your custom list item layout xml
- Constructor:
  1. Context ← activity instance
  2. Resource Id of a view for the list item
  3. Resource Id of a TextView in the list item layout
  4. The array or list of data
- Override getView(int, View, ViewGroup)

Hands-on Coding

Walk through **SampleList.java**

1. A simple list - use android default layout
2. A pretty list - you can style the item
3. A fancy list - use custom ArrayAdapter

Break

Fragment

What is a fragment?

- a component of UI in an activity
- must exist as part of an activity

Why do we need it?

- introduced in Honeycomb to better support tablet and phone UI
- dynamic and flexible UI

UI Patterns with Fragments

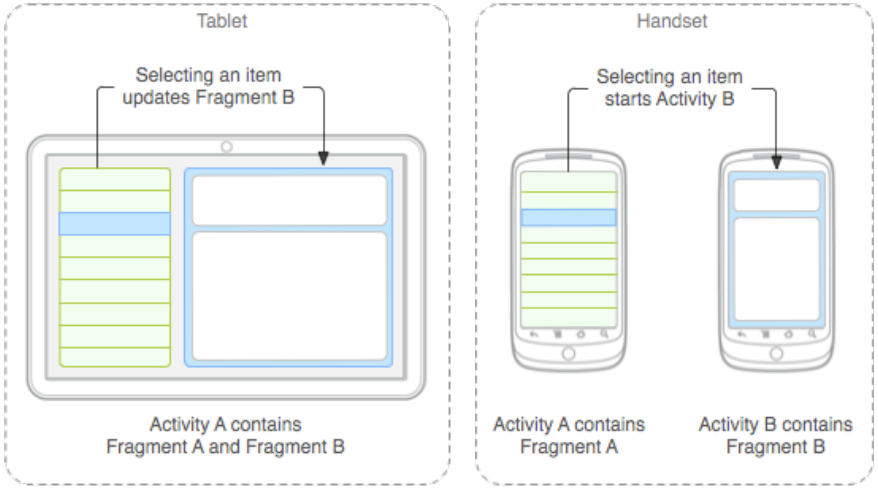
Fragments are used everywhere!

- Master/Detail list
- Swipe View w/ Tabs
- ViewPager
- Navigation Drawer

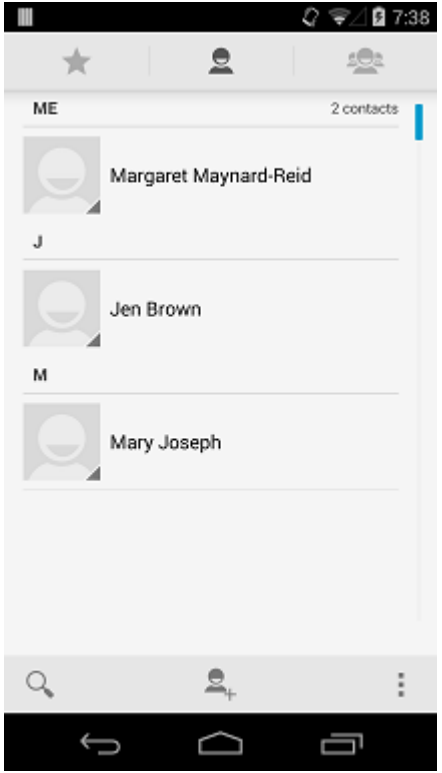
Master/Detail List

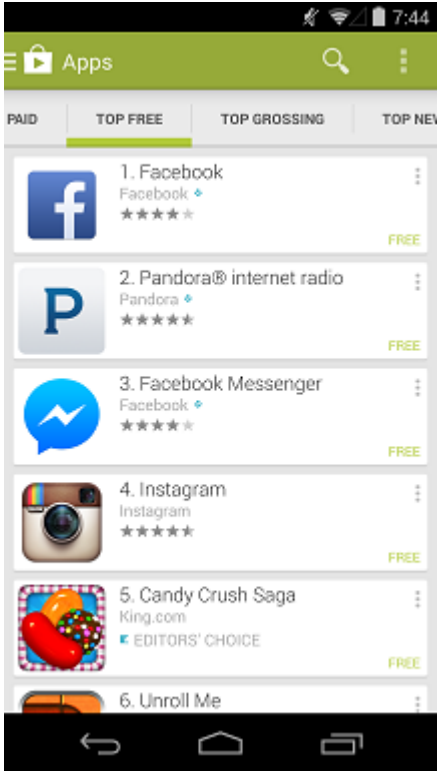
This is particularly useful to create UI optimized for both phone and tablet.

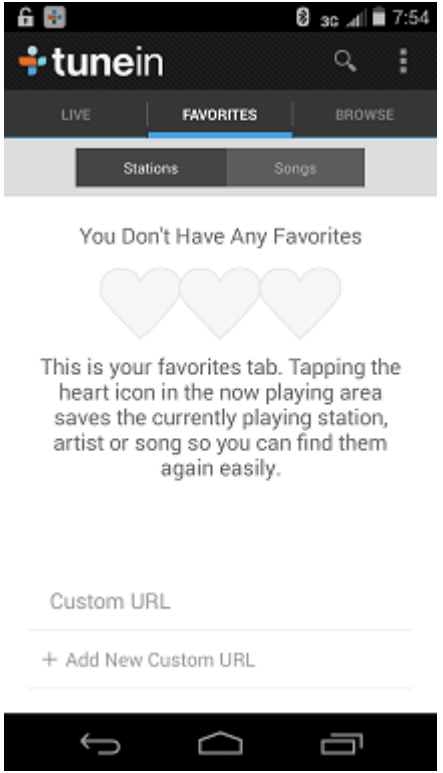
- Single-pane: on phone & tablet portrait mode
- Two-pane: on tablet landscape mode



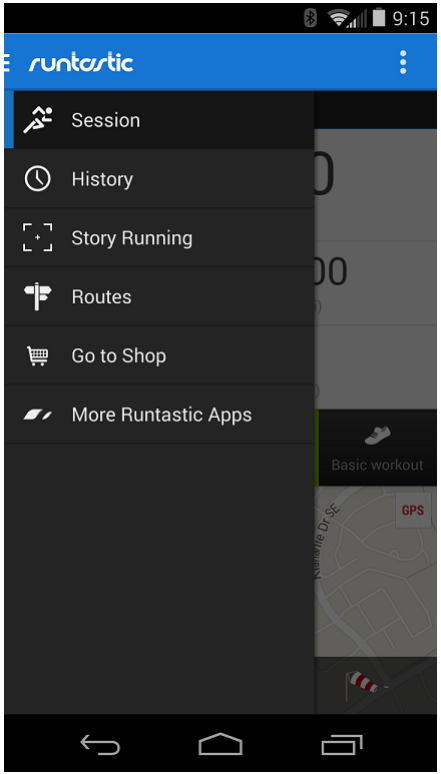
Swipe View w/ Tabs & ViewPager

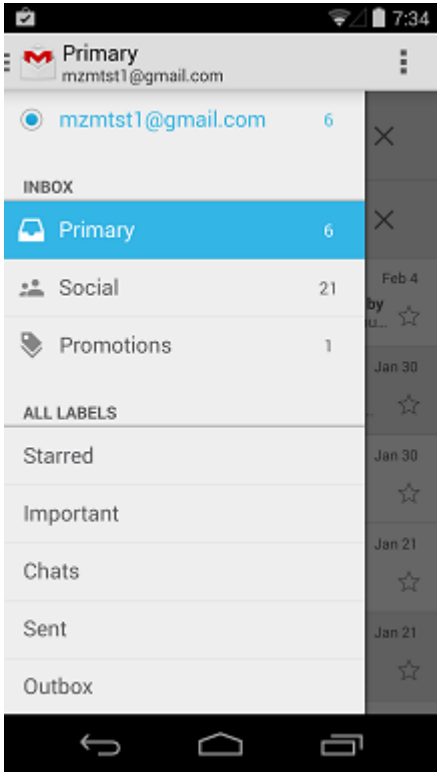


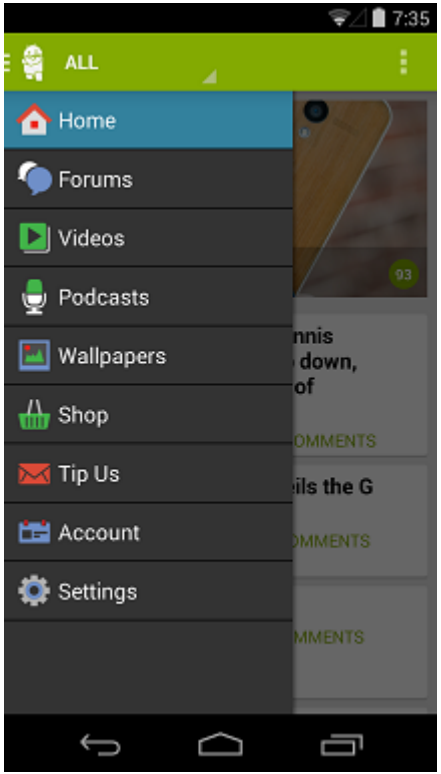




Navigation Drawer







Android Fragment Key Classes

- **Fragment -** base class for creating a fragment
- **FragmentManager -** interface that allows you to interact with the fragment
- **FragmentTransaction -** API that performs fragment operations: add, remove and replace etc.

Android Fragment Subclasses

- DialogFragment
- ListFragment
- PreferenceFragment
- WebViewFragment
- MapFragment

Creating a Fragment

1. Define a **layout** called example_fragment.xml
2. Extend the ***Fragment*** class or the subclasses
3. Must implement ***onCreateView*** method, & return a view for the fragment

```
public static class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.example_fragment, container, false);
    }
}
```

** No need to register fragment in AndroidManifest.xml

Adding a Fragment

- Fragment can’t exist on its own, and must be part of an activity.
- Two ways to add a fragment to an activity:
  1. by XML layout statically
  2. by code programmatically

Add a Fragment by XML Layout

Use this approach if the fragment is fairly static

Add fragment to activity layout using the “fragment” tag

The ***android:name*** attribute specifies the **Fragment** class to instantiate in the **activity l**ayout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="horizontal"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <fragment android:name="com.example.ArticleListFragment"
         android:id="@+id/list"
         android:layout_width="wrap_content"
         android:layout_height="match_parent" />
</LinearLayout>
```

Add a Fragment By Code

Use this approach if you would like to **dynamically add/remove/replace fragments to your activity** at runtime:

Step 1. Define container(s) in the activity layout for the fragment(s):

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Framelayout
        android:id="@+id/fragment_container"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent" />
</LinearLayout>
```

Add a Fragment By Code

Step 2. In the activity, use FragmentManager and FragmentTransaction to add a fragment

```
FragmentManager fragmentManager = getFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

Sample Code

Walk through **SampleFragment.java**

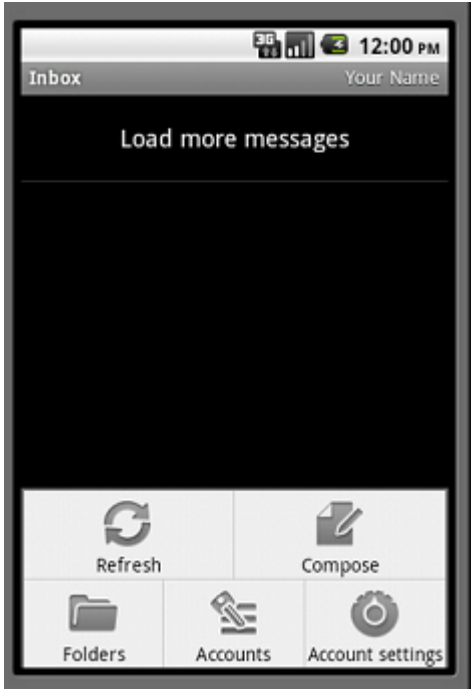
- Create a fragment
- Add a fragment statically
- Add a fragment dynamically by code

Break

Options Menu

History 2010 - 2011:

- A menu rises up from bottom of screen
- Launched by a MENU key



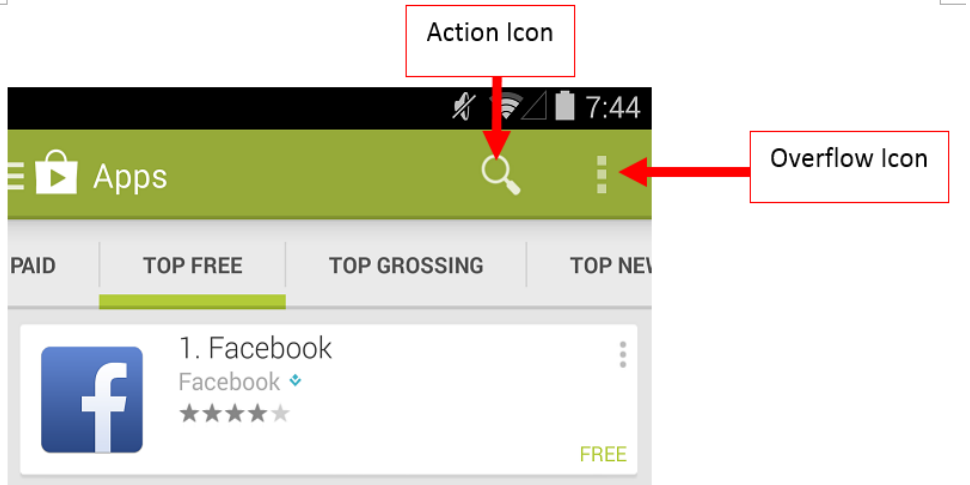
Action Bar

- Introduced in Android 3.0 (API 11)
- A dedicated real estate for your brand
- Provide user with navigation and access to actions
- For backwards compatibility, add support library with resources (v7 appcompat library). We used to use ActionBarSherlock.

Action Bar - menu items

- Add action icons / menu items
- The overflow menu

android:showAsAction=["ifRoom" | "never" | "withText" | "always"]



Action Bar - “Up” Affordance

Step1. set action bar setDisplayHomeAsUpEnabled as true

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_details);
    ActionBar actionBar = getActionBar();
    actionBar.setDisplayHomeAsUpEnabled(true);
    ...
}
```

Action Bar - “Up” Affordance

Step 2. Specify parent activity in AndroidManifest.xml:

```
<!-- A child of the main activity -->
<activity
    android:name="com.example.myfirstapp.DisplayMessageActivity"
    android:label="@string/title_activity_display_message"
    android:parentActivityName="com.example.myfirstapp.MainActivity" >
    <!-- Parent activity meta-data to support API level 7+ -->
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.example.myfirstapp.MainActivity" />
</activity>
```

Action Bar - Show/Hide it

- Included by default with Theme.holo, targetSdkVersion or minSdkVersion => API11
- You can show/hide the ActionBar

```
ActionBar actionBar = getActionBar();
actionBar.hide();
actionBar.show();
```

Action Bar - Customize its Look

- Use Android Studio to style app icon:

  http://android-ui-utils.googlecode.com/hg/asset-studio/dist/icons-launcher.html
- Use a color that matches your brand or use the Android colors:

  http://developer.android.com/design/style/color.html

Customize ActionBar background

In styles.xml

```
<!-- Base application theme. -->
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    <!-- Customize my actionBar -->
    <item name="actionBarStyle">@style/MyActionBar</item>
</style>

<!-- actionBar style, with customized background color -->
<style name="MyActionBar" parent="@style/Widget.AppCompat.Light.ActionBar">
    <!-- set background color -->
    <item name="background">@color/blue</item>
</style>
```

Sample Code

Walk through **SampleActionBar.java**

- App icon on ActionBar
- Customize ActionBar background

App Navigation Best Practices

- Use ActionBar icons for actions (i.e. add, change, share, search etc.)
- Use Nav Drawer if your app has a complex and deep hierarchy.
- Use Tabs or ViewPager to switch between similar pages.

Appendix

- **Fragments**

  http://developer.android.com/guide/components/fragments.html
- **ActionBar**

  http://developer.android.com/guide/topics/ui/actionbar.html
- **Android Design Guidelines**

  https://developer.android.com/design/index.html
- **Android Design in Action: Navigation Anti-Patterns**

  http://www.youtube.com/watch?v=Sww4omntVjs