Unit 3
Application Lifecycle & Intents

Alexander Lucas and Richard Hyndman
Android Developer Advocates,
14-August-2012

Unit 3 - Topics

Understanding Application Lifecycle

- Application Types
- Fundamental Components
- Handling State Changes

System Messaging using Intents

- Sending Explicit, Implicit and Broadcast Messages
- Handling messages

Using Background Services

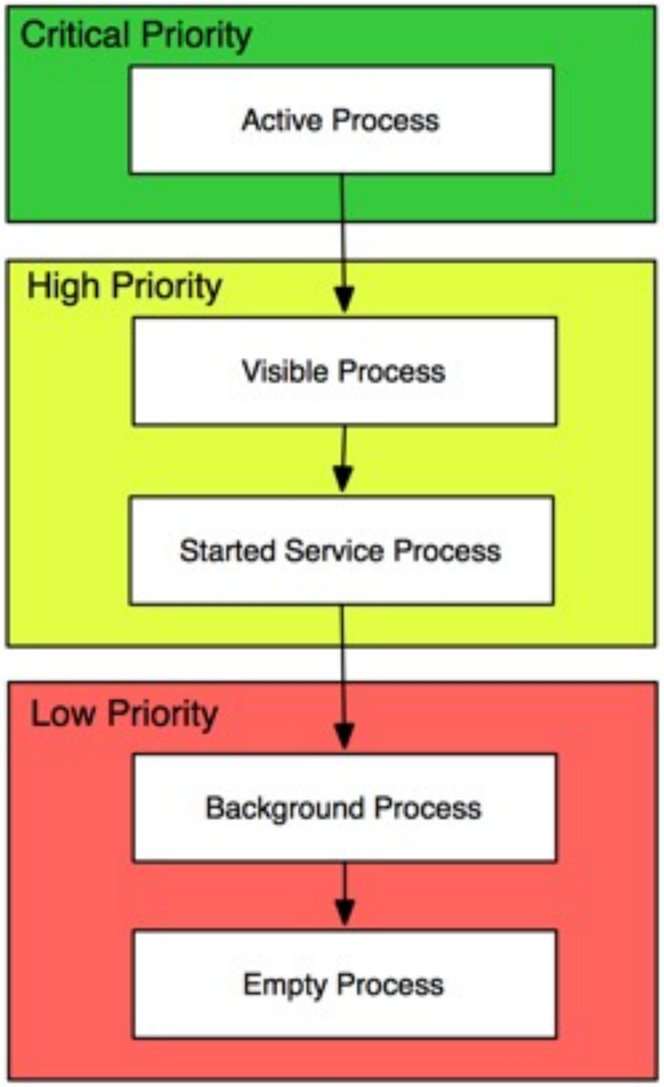
- Long-running background process
- No UI

Understanding Application Lifecycle

Application Lifecycle

- Apps have Limited Control
  - they are ***managed by the system***
  - system aggressively manages resources for stability
  - components listen for changes in state and react
- Processes & Memory
  - each app runs within a separate process
  - each process runs a separate Dalvik instance
  - process and mem management handled by the system
  - processes and their hosted apps are killed without warning
  - must handle unexpected termination

Application Priority



Lifetime by Priority

- Android kills processes and their hosted app to recover system resources
- Termination order determined by hosted application priority
- Two at the same priority, longest running is killed first

Types of Applications

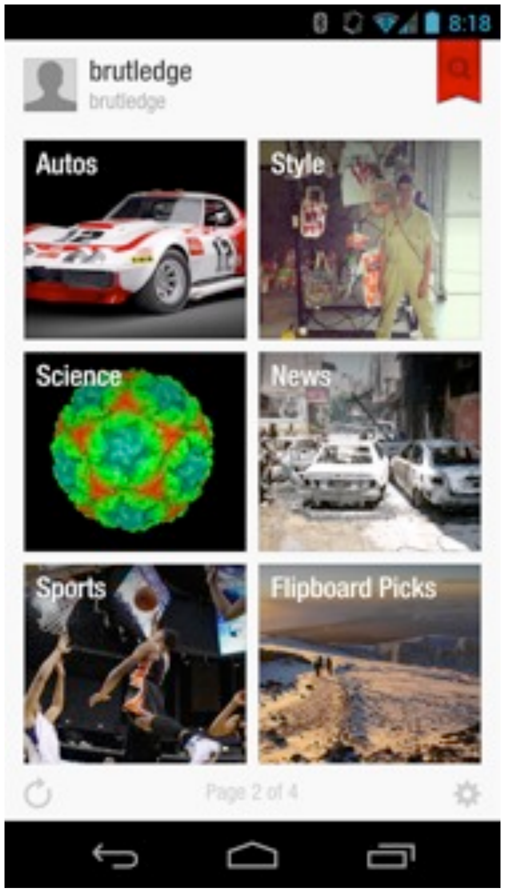
- Foreground
  - usable in the foreground, otherwise suspended
  - *example*: casual games
- Background
  - little user experience beyond config
  - *example*: alarm clock
- Intermittent
  - have a robust UI and do work in the background
  - *example*: news app, email
- Widget & Live Wallpaper
  - represented only on the homescreen
  - *example*: battery level widget

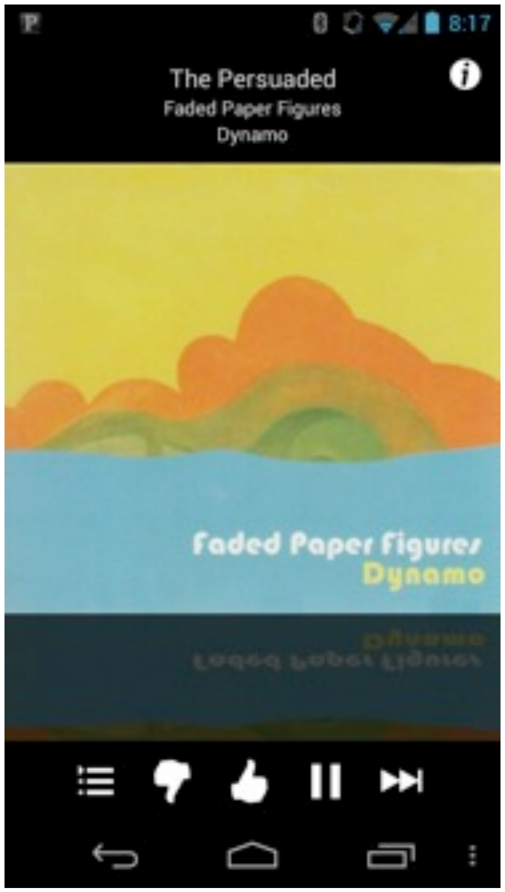
Fundamental Components

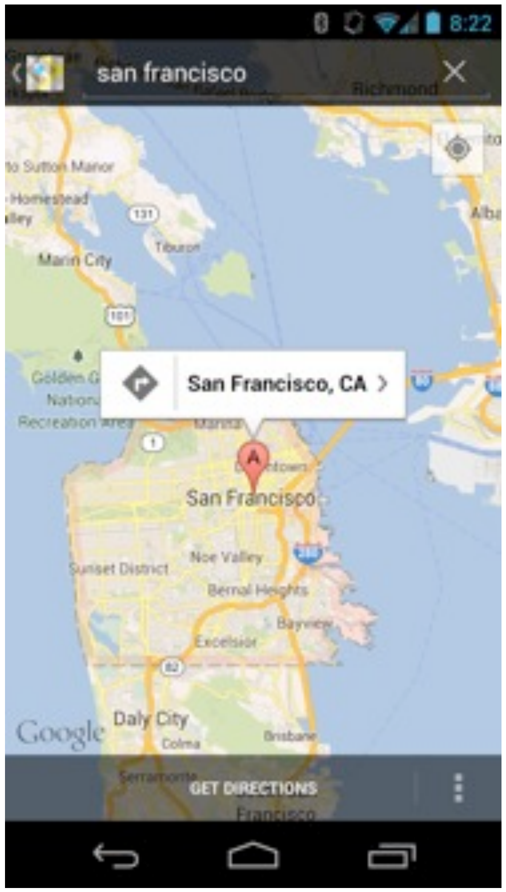
- Activity
  - represents a screen or "activity of use" to the user
  - handles lifecycle events from the system
- Intent
  - system message used for data interchange
  - used to invoke other components
- BroadcastReceiver
  - system listeners to handle system messages
- Service
  - background process
  - runs independantly of when your application is in use.
- *ContentProvider – covered in later unit*

The Activity Class

- Represents a screen, presented to a user
- UI separated into a '*layout resource*' xml file
- Assign the UI by calling `setContentView()`







The Activity Class

- Created by the system, then call to onCreate(…)
- UI layout is assigned using setContentView(…)

```
import android.app.Activity;
import android.os.Bundle;

public class ActivityA extends Activity {

    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_a);
    }
}
```

The Android Manifest File

- xml file, describes application and components
- bundled within your .apk file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.google.androidcamp.unit3"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="16" />

    <application android:label="@string/app_name">
        <activity android:name=".ActivityA"
                  android:label=”My Application">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
     </application>
</manifest>
```

Let’s try it!

Requirements

Basic Development Environment

1. Java JDK 1.6
2. Ant build tool
   code.google.com/p/main-gac2012/downloads
3. Simple text editor
4. Android Debug Bridge (ADB)
5. Access to terminal window

Consider IntelliJ by JetBrains

Awesome Java IDE

1. Killer Android support
2. Community Version is free, open source
3. Fast, intuitive with great refactoring and debugging

http://www.jetbrains.com/idea/download

Setting Up Ant

Follow these steps:

1. Download ant.zip (version 1.8.x or better)
2. Unzip in your home dir
3. Update system path to include `<ANT_HOME>/bin`
4. Verify by running command `ant -version`

```
brutledge$ ant -version
Apache Ant(TM) version 1.8.2 compiled on June 3 2011
```

Setting it Up Android Device

Follow these steps:

1. Update system path to include SDK home
2. Verify by running command `android`
3. Connect Nexus 7 device to USB cable
4. Verify by running command `adb devices`

```
brutledge$ adb devices
List of devices attached
016B756E02010016 device
```

Code Exercise 1

Goal: Create a single Activity application

1. Create a new Android project
2. Edit `ActivityA.java` and log the lifecycle event
3. Compile, install and run

1. Create a new Android project

Use the ‘android’ command

```
brutledge$ android create project --target 16 --path ./
--package com.example.unit3.excercise1 --activity ActivityA
```

- creates a fully structured project
- creates a skeleton Activity class
- creates AndroidManifest.xml file
- creates an ant build.xml file

1. Create a new Android project

What did it create?

```
brutledge$ ls -lart
total 48
drwxr-xr-x  18 brutledge  5000   612 Jul 21 20:38 ..
drwxr-xr-x   3 brutledge  5000   102 Jul 21 20:38 src
drwxr-xr-x   4 brutledge  5000   136 Jul 21 20:38 res
-rw-r--r--   1 brutledge  5000   563 Jul 21 20:38 project.properties
-rw-r--r--   1 brutledge  5000   781 Jul 21 20:38 proguard-project.txt
-rw-r--r--   1 brutledge  5000   429 Jul 21 20:38 local.properties
drwxr-xr-x   2 brutledge  5000    68 Jul 21 20:38 libs
-rw-r--r--   1 brutledge  5000  3921 Jul 21 20:38 build.xml
drwxr-xr-x   2 brutledge  5000    68 Jul 21 20:38 bin
-rw-r--r--   1 brutledge  5000   698 Jul 21 20:38 ant.properties
-rw-r--r--   1 brutledge  5000   606 Jul 21 20:38 AndroidManifest.xml
drwxr-xr-x  12 brutledge  5000   408 Jul 21 20:38 .
```

2. Edit class ActivityA

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class ActivityA extends Activity {

    private static final String TAG = "ANDROID_CAMP";
    private static final String NAME = "ActivityA";

    /** Called when the activity is first created */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.i(TAG, NAME + " created");
        setContentView(R.layout.main);
    }
}
```

3. Compile, install and run

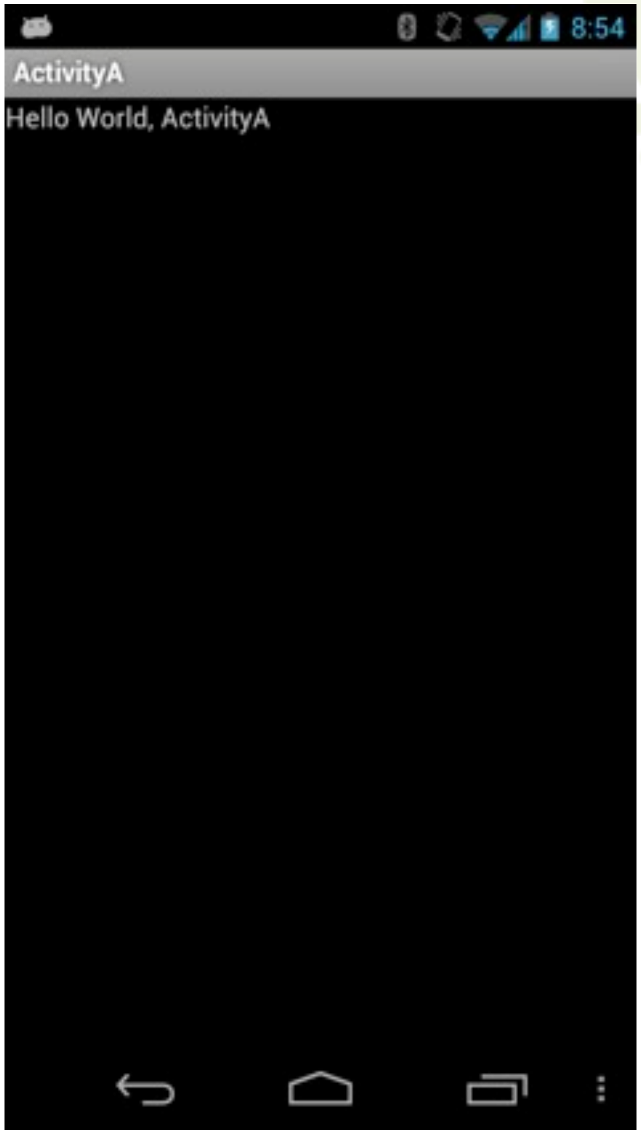
Use the ‘ant’ command

```
brutledge$ ant debug install
Buildfile: /Users/brutledge/workspace/android-camp/build.xml

        ...

install:
     [echo] Installing /Users/brutledge/workspace/android-camp/bin/
ActivityA-debug.apk onto default emulator or device...
     [exec] 784 KB/s (4829 bytes in 0.006s)
     [exec] * daemon not running. starting it now on port 5037 *
     [exec] * daemon started successfully *
     [exec] 	pkg: /data/local/tmp/ActivityA-debug.apk
     [exec] Success

BUILD SUCCESSFUL
Total time: 28 seconds
```

Success! A Single Activity App

Use ‘adb logcat’ command

```
brutledge$ adb logcat -s "ANDROID_CAMP"
--------- beginning of /dev/log/system
--------- beginning of /dev/log/main
I/ANDROID_CAMP(19061): ActivityA created
```



Hello World, ActivityA

Activity States

- Active
  - visible, focused and receiving user input
  - *critical priority*: it's what the user is currently doing
- Paused
  - visible (often partially visible), does not have focus
  - stop ongoing actions that shouldn't continue
  - *example*:   stop video, stop animations, persist unsaved data
  - *high priority:* not likely to be terminated and recycled

Activity States

- Stopped
  - not visible, remains in memory
  - using ‘recent apps’ switcher will stop your Activity
  - release all resources that aren’t needed
  - *example*: write to a database and release connections
  - *medium priority*: candidate for termination and recycle
- Inactive
  - after it has been killed, or before it has been started
  - no longer in the Activity stack
  - *low priority*: will be recycled

Activity Lifecycle Methods

- exposes lifecycle event handler methods
- fired when Activity changes states reacting to system

State	Startup Method	Teardown Method
Active	onCreate()	onDestroy()
Paused	onPause()	onResume()
Stopped	onStart()	onStop()

Let’s try it!

Code Exercise 2

Goal: Observe Activity Lifecycle Methods

1. Edit `ActivityA.java` and log all lifecycle events
   - For easy reference, Activity Javadocs are at:
     - http://developer.android.com/reference/android/app/Activity.html
     - Or just developer.android.com and search for “Activity”.
     - Get good at that. No, really.

2. Compile, install and run

1. Edit class ActivityA

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class ActivityA extends Activity {
   ...

    @Override
    protected void onStart() {
        super.onStart();
        Log.i(TAG, NAME + " started");
    }
   ...
}
```

Override all lifecycle methods:

onStart(), onStop(), onPause(), onResume(), onDestroy()

2. Compile, install and run

Use the ‘ant’ command

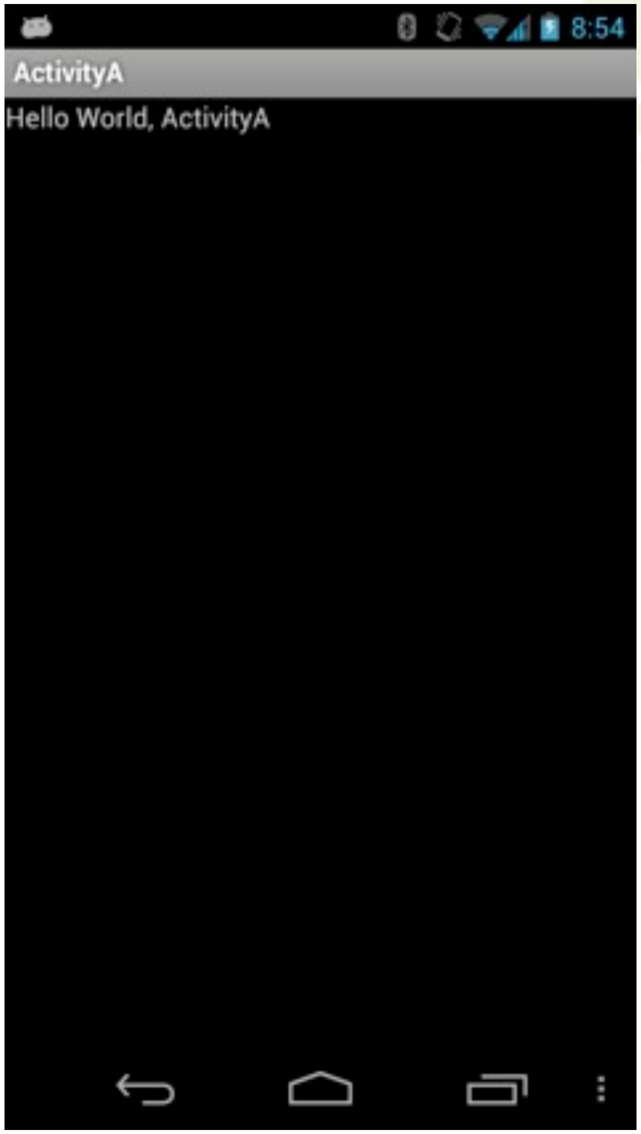
```
brutledge$ ant debug install
Buildfile: /Users/brutledge/workspace/android-camp/build.xml

          ...

install:
     [echo] Installing /Users/brutledge/workspace/android-camp/bin/
ActivityA-debug.apk onto default emulator or device...
     [exec] 784 KB/s (4829 bytes in 0.006s)
     [exec] * daemon not running. starting it now on port 5037 *
     [exec] * daemon started successfully *
     [exec]    pkg: /data/local/tmp/ActivityA-debug.apk
     [exec] Success

BUILD SUCCESSFUL
Total time: 28 seconds
```

Success! Logging lifecycle events

`adb logcat -s “ANDROID_CAMP”`

- **Launch app, then hit back button**
- **Launch app, then hit home button**
- **Notice a difference?**

```
brutledge$ adb logcat -s "ANDROID_CAMP"
--------- beginning of /dev/log/system
--------- beginning of /dev/log/main
I/ANDROID_CAMP(13777): ActivityA created
I/ANDROID_CAMP(14651): ActivityA started
I/ANDROID_CAMP(14651): ActivityA resumed
I/ANDROID_CAMP(14651): ActivityA paused
I/ANDROID_CAMP(14651): ActivityA stopped
I/ANDROID_CAMP(14651): ActivityA destroyed
```



Hello World, ActivityA

Saving State

- Apps should be resilient, easily recover upon destruction
- Activity provides additional lifecycle methods

`onSaveInstanceState(Bundle bundle)`

- called after onStop(), when the Activity is killed by the system

`onRestoreInstanceState(Bundle bundle)`

- called after onStart(), when the Activity was killed by the system

Let’s try it!

Code Exercise 3

Goal: Observe Saved Instance State

1. Download android-camp-unit-3.zip
   code.google.com/p/main-gac2012/downloads
2. Unzip in working directory
3. Open /android-camp/unit-3/exercise-3
4. Update `local.properties` with SDK path

Notice resource files

res/values/colors.xml

```
<resources>
    <color name="white">#FFFFFF</color>
    <color name="black">#000000</color>

    <color name="light_blue">#A8DFF4</color>
    <color name="light_green">#D3E992</color>
    <color name="light_red">#FFAFAF</color>
    <color name="light_yellow">#FFECC0</color>

    <color name="dark_blue">#0099CC</color>
    <color name="dark_green">#669900</color>
    <color name="dark_red">#CC0000</color>
    <color name="dark_yellow">#FF8A00</color>
</resources>
```

Notice resource files

res/values/dimensions.xml

```
<resources>
    <dimen name="font_large">44dp</dimen>
    <dimen name="font_medium">24dp</dimen>
    <dimen name="font_small">10dp</dimen>
</resources>
```

Notice resource files

res/values/strings.xml

```
<resources>
    <string name="app_name">Android Camp</string>
    <string name="lbl_activity_a">Activity A</string>
    <string name="lbl_activity_b">Activity B</string>
    <string name="btn_start_a">Start Activity A</string>
    <string name="btn_start_a">Start Activity B</string>
</resources>
```

Extra Credit

- Translate to another language

res/**values-es**/strings.xml

Notice class ActivityA

Override onSaveInstanceState(…)

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class ActivityA extends Activity {
   ...

    @Override
    public void onSaveInstanceState(Bundle bundle) {
        super.onSaveInstanceState(bundle);
        Log.i(TAG, NAME + " onSaveInstanceState");
        bundle.putString(”foo", ”bar”);
    }

    ...
}
```

Notice class ActivityA

Override onRestoreInstanceState(…)

```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class ActivityA extends Activity {
   ...

    @Override
    public void onRestoreInstanceState(Bundle bundle) {
        super.onRestoreInstanceState(bundle);
        Set<String> keys = bundle.keySet();
        if (keys.size() > 0) {
            for (String key : keys) {
                Log.i(TAG, key + ", " + bundle.get(key));
            }
        }
    }
    ...
}
```

Compile, install and run

Use the ‘ant’ command

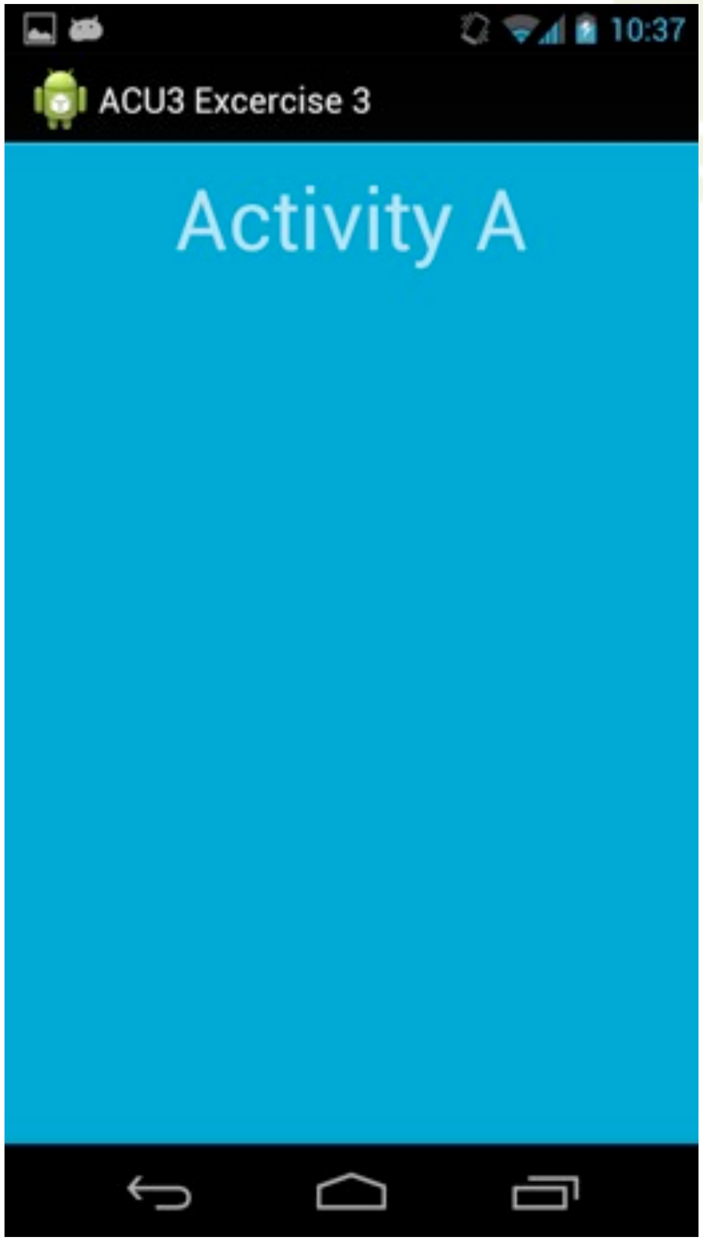
```
brutledge$ ant debug install
Buildfile: /Users/brutledge/workspace/android-camp/build.xml

         ...

install:
     [echo] Installing /Users/brutledge/workspace/android-camp/bin/
ActivityA-debug.apk onto default emulator or device...
     [exec] 784 KB/s (4829 bytes in 0.006s)
     [exec] * daemon not running. starting it now on port 5037 *
     [exec] * daemon started successfully *
     [exec]    pkg: /data/local/tmp/ActivityA-debug.apk
     [exec] Success

BUILD SUCCESSFUL
Total time: 28 seconds
```

Success! Logging lifecycle events

```
adb logcat -s "ANDROID_CAMP"
```

- Launch app
- Rotate device

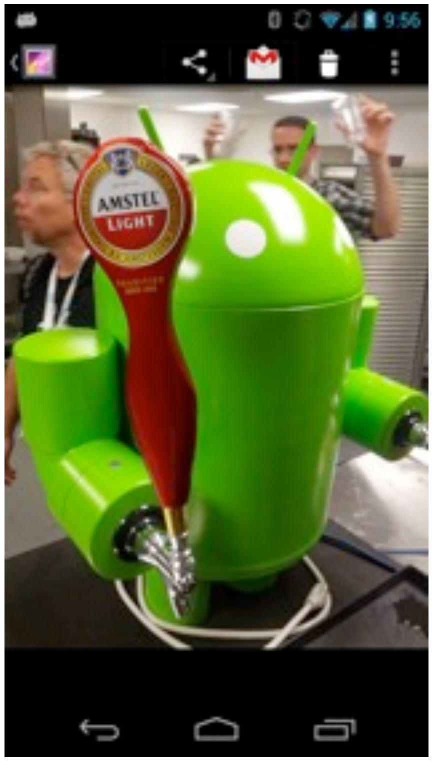
```
I/ANDROID_CAMP( 6862): ActivityA created
I/ANDROID_CAMP( 6862): ActivityA started
I/ANDROID_CAMP( 6862): ActivityA resumed
I/ANDROID_CAMP( 6862): ActivityA paused
I/ANDROID_CAMP( 6862): ActivityA onSaveInstanceState
I/ANDROID_CAMP( 6862): ActivityA stopped
I/ANDROID_CAMP( 6862): ActivityA destroyed
I/ANDROID_CAMP( 6862): ActivityA created
I/ANDROID_CAMP( 6862): ActivityA started
I/ANDROID_CAMP( 6862): ActivityA onRestoreInstanceState
I/ANDROID_CAMP( 6862): ActivityA foo, bar
I/ANDROID_CAMP( 6862): ActivityA resumed
```

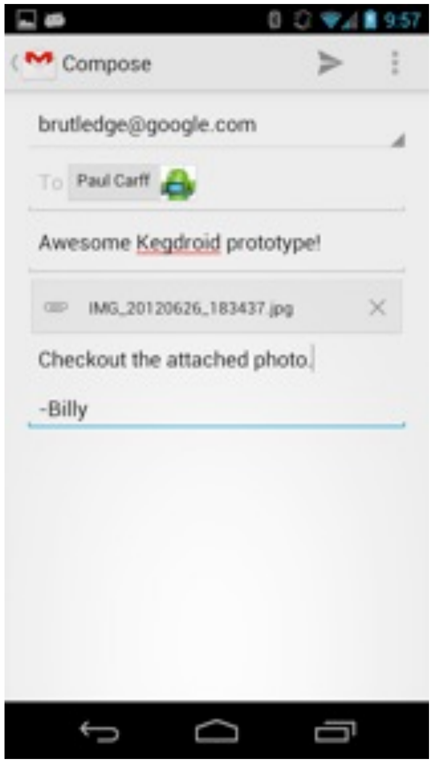


System Messaging & Intents

System Messaging

- Apps are loosely coupled components
- Bound together by system messages or “*Intents*”
- Allows for replacement of application elements
- Allows applications to interact and share features




Compose

brutledge@google.com

To Paul Carff

Awesome Kegdroid prototype!

IMG_20120626_183437.jpg

Checkout the attached photo.

-Billy

Different Intent Types

- Explicit Intent
  - start new Activity by explicit class name
  - within your application or other
  - *example*: ActivityA starts ActivityB
- Implicit Intent
  - requesting an action be performed
  - handled by any qualified, registered application
  - *example*: request an app to handle sharing a photo
- Broadcast Intent
  - broadcast events to the entire system
  - create BroadcastReceivers as handlers
  - *example*: low battery event, power connected event

The Intent Class

- Constructed for explicit messaging

```
// Send an explicit intent to start ActivityB
Intent intent = new Intent(ActivityA.this, ActivityB.class);
startActivity(intent);
```

- Constructed for implicit messaging

```
// Send an implicit intent to dial a phone number
String action = Intent.ACTION_DIAL;
Uri uri = Uri.parse(”tel:415-555-1212”);
Intent intent = new Intent(Intent.ACTION_DIAL, uri);
startActivity(intent);
```

The Activity Stack

- Last-in-first-out collection of Activities
- Activity state determined by position in the stack
- System uses stack to measure priority

Let’s try it!

Code Exercise 4

Goal: Explicitly Launch ActivityB

1. Download android-camp-unit-3.zip
   code.google.com/p/main-gac2012/downloads
2. Unzip in working directory
3. Open /android-camp/unit-3/exercise-4
4. Update `local.properties` with SDK path

Notice resource files

res/layout/activity_a.xml

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:background="@color/dark_blue"
              android:padding="8dip">

    <TextView android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:text="@string/lbl_activity_a"
              android:gravity="center_horizontal"
              android:textSize="@dimen/font_large"
              android:textColor="@color/light_blue"
              android:paddingBottom="16dip"/>

    <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
                    android:orientation="vertical"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center_horizontal">

        <Button android:id="@+id/btn_start_b"
                android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:text="@string/btn_start_b"
                android:onClick="startActivityB" />

    </RelativeLayout>

</LinearLayout>
```

Notice resource files

res/layout/activity_b.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:background="@color/dark_yellow"
              android:padding="8dip">
    <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/lbl_activity_b"
            android:gravity="center_horizontal"
            android:textSize="@dimen/font_large"
            android:textColor="@color/light_yellow"
            android:paddingBottom="16dip"/>

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                  android:orientation="horizontal"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_gravity="center_horizontal">

        <Button android:id="@+id/btn_finish"
                android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:text="@string/btn_finish_b"
                android:onClick="finishActivityB" />
    </LinearLayout>

</LinearLayout>
```

Notice resource files

res/**layout-land**/activity_b.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:background="@color/dark_yellow"
              android:padding="8dip">
    <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/lbl_activity_b"
            android:gravity="center_horizontal"
            android:textSize="@dimen/font_large"
            android:textColor="@color/light_yellow"
            android:paddingBottom="16dip"/>

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                  android:orientation="horizontal"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_gravity="center_horizontal">

        <Button android:id="@+id/btn_finish"
                android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:text="@string/btn_finish_b"
                android:onClick="finishActivityB" />
    </LinearLayout>

</LinearLayout>
```

Notice change to ActivityA

```
package com.google.androidcamp.unit3.excercise4;

public class ActivityA extends Activity {

    private static final String TAG = "ANDROID_CAMP";
    private static final String NAME = "ActivityA";

    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        Log.i(TAG, NAME + " created");
        setContentView(R.layout.activity_a);
    }

    ...

    public void startActivityB(View view) {
        Intent intent = new Intent(ActivityA.this, ActivityB.class);
        startActivity(intent);
    }
}
```

Notice new ActivityB

```
package com.google.androidcamp.unit3.excercise4;

public class ActivityB extends Activity {

    private static final String TAG = "ANDROID_CAMP";
    private static final String NAME = "ActivityB";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.i(TAG, NAME + " created");
        setContentView(R.layout.activity_b);
    }

    ...

    public void finishActivityB(View view) {
        finish();
    }

}
```

Notice explicit Intent

Add explicit intent to start `ActivityB` from `ActivityA`

```
public void startActivityB(View view) {
    Intent intent = new Intent(ActivityA.this, ActivityB.class);
    startActivity(intent);
}
```

Associate with button click handler in `activity_a.xml`

```
<Button android:id="@+id/btn_start_b"
                android:layout_height="wrap_content"
                android:layout_width="wrap_content"
                android:text="@string/btn_start_b"
                android:onClick="startActivityB" />
```

… same with button click handler in `activity_b.xml`

Notice AndroidManifest.xml

Declare `ActivityB` in the manifest

```
<activity android:name=".ActivityB"
          android:label="@string/app_name">
</activity>
```

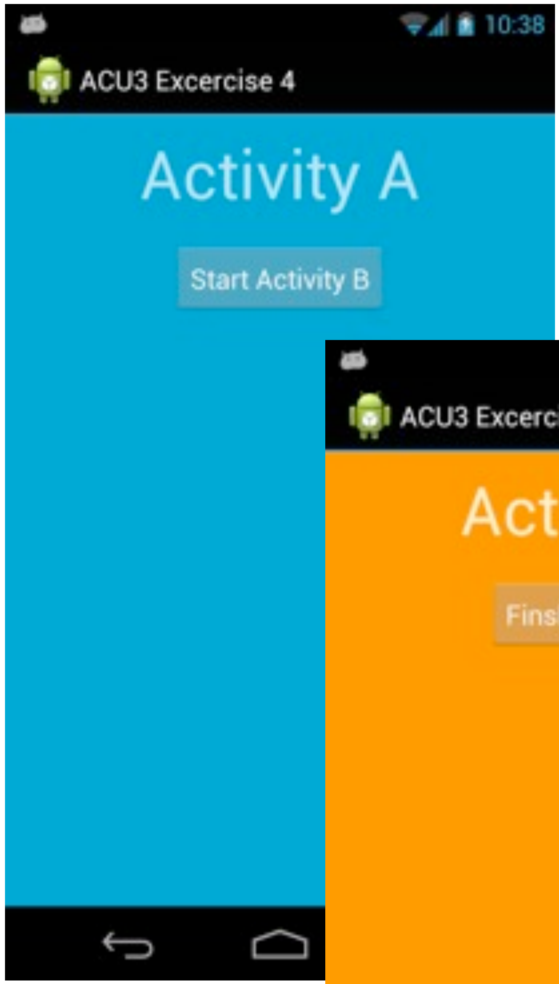
Compile, install and run

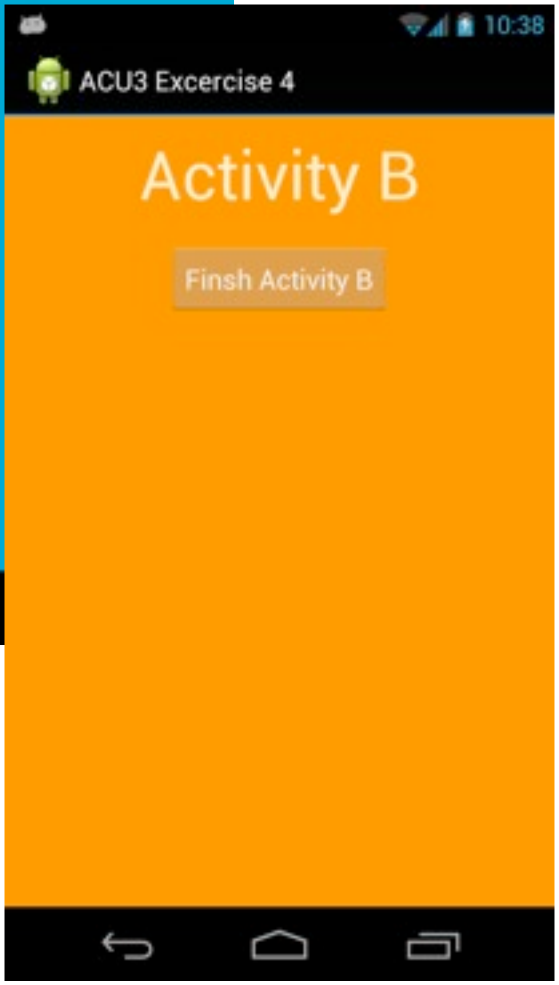
Use the ‘ant’ command

```
brutledge$ ant debug install
Buildfile: /Users/brutledge/workspace/android-camp/build.xml

         ...

install:
     [echo] Installing /Users/brutledge/workspace/android-camp/bin/
ActivityA-debug.apk onto default emulator or device...
     [exec] 784 KB/s (4829 bytes in 0.006s)
     [exec] * daemon not running. starting it now on port 5037 *
     [exec] * daemon started successfully *
     [exec]    pkg: /data/local/tmp/ActivityA-debug.apk
     [exec] Success

BUILD SUCCESSFUL
Total time: 28 seconds
```

Success! Explicit Launch of ActivityB

```
brutledge$ adb logcat -s "ANDROID_CAMP"
--------- beginning of /dev/log/system
--------- beginning of /dev/log/main
I/ANDROID_CAMP(15640): ActivityA created
I/ANDROID_CAMP(15640): ActivityA started
I/ANDROID_CAMP(15640): ActivityA resumed
I/ANDROID_CAMP(15640): ActivityA paused
I/ANDROID_CAMP(15640): ActivityB created
I/ANDROID_CAMP(15640): ActivityB started
I/ANDROID_CAMP(15640): ActivityB resumed
I/ANDROID_CAMP(15640): ActivityA stopped
I/ANDROID_CAMP(15640): ActivityB paused
I/ANDROID_CAMP(15640): ActivityB stopped
```





Returning Results

No Results Required

`startActivity(…)`

– launched Activity ***does not*** provide any feedback when it closes

To Handle Results

`startActivityForResult(…)`

– additional parameter for ***request code***

`onActivityResult(…)`

– override this method to evaluate the ***result code***

Let’s try it!

Monday, August 13, 12

Code Exercise 5

Goal: Return Result from ActivityB

1. Open /android-camp/unit-3/exercise-5
2. Update `local.properties` with SDK path

Notice update to ActivityA

```
package com.google.androidcamp.unit3.excercise5;

public class ActivityA extends Activity {

   private static final int REQUEST_CODE = 1;

   public void startActivityB(View view) {
        Intent intent = new Intent(ActivityA.this, ActivityB.class);
        startActivityForResult(intent, REQUEST_CODE);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        switch (requestCode) {
            case (REQUEST_CODE):
                if (resultCode == Activity.RESULT_OK)
                    Log.i(TAG, NAME + " result returns ok");
                    // evaluate data further here if desired
                else if (resultCode == Activity.RESULT_CANCELED)
                    Log.i(TAG, NAME + " result returns canceled");
                    // evaluate data further here if desired
                else
                    Log.i(TAG, NAME + " result code=" + resultCode);
                    // evaluate data further here if desired
                break;
        }
    }
}
```

Notice update to ActivityB

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Log.i(TAG, NAME + " created");
    setContentView(R.layout.activity_b);

    Button btnOk = (Button)findViewById((R.id.btn_ok));
    btnOk.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
          Intent result = new Intent();
          setResult(RESULT_OK, result);
          finish();
        }
    });

    Button btnCancel = (Button)findViewById((R.id.btn_cancel));
    btnCancel.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
            Intent result = new Intent();
            setResult(RESULT_CANCELED, result);
            finish();
        }
    });
}
```

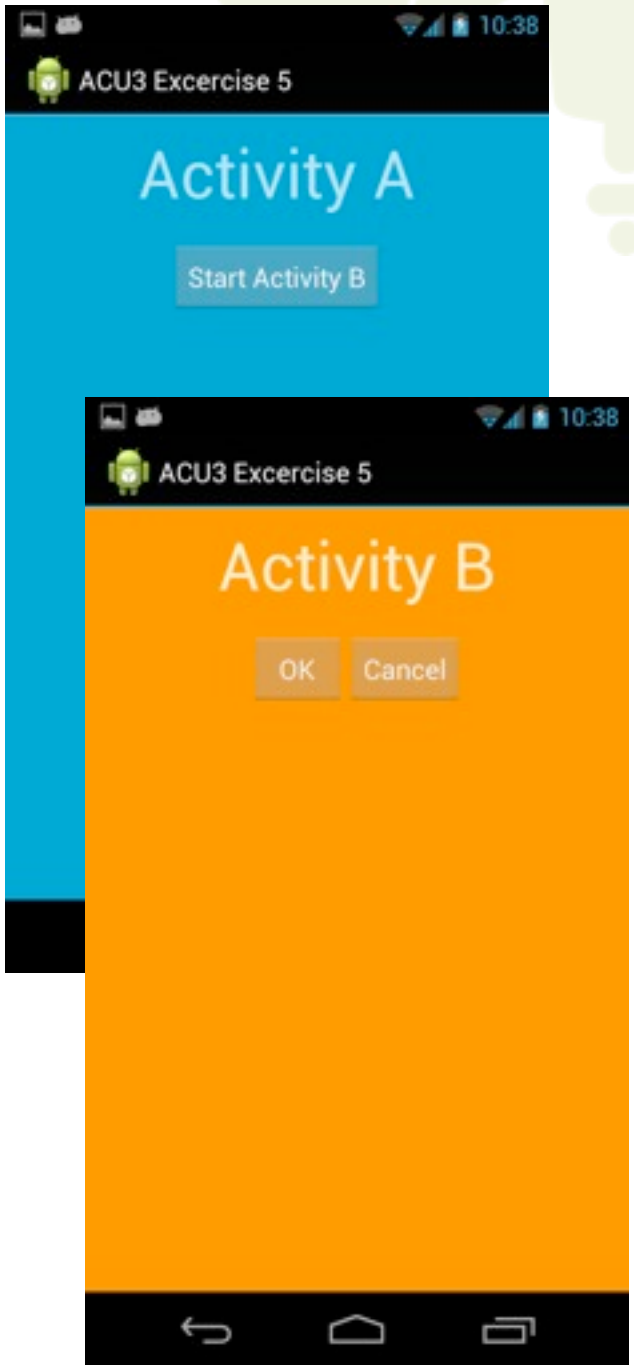
Compile, install and run

Use the ‘ant’ command

```
brutledge$ ant debug install
Buildfile: /Users/brutledge/workspace/android-camp/build.xml

          ...

install:
     [echo] Installing /Users/brutledge/workspace/android-camp/bin/
ActivityA-debug.apk onto default emulator or device...
     [exec] 784 KB/s (4829 bytes in 0.006s)
     [exec] * daemon not running. starting it now on port 5037 *
     [exec] * daemon started successfully *
     [exec]    pkg: /data/local/tmp/ActivityA-debug.apk
     [exec] Success

BUILD SUCCESSFUL
Total time: 28 seconds
```

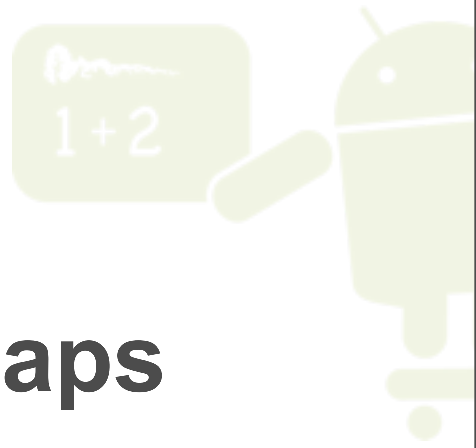
Success! Returning Results

```
brutledge$ adb logcat -s "ANDROID_CAMP"
--------- beginning of /dev/log/system
--------- beginning of /dev/log/main
I/ANDROID_CAMP(29099): ActivityA created
I/ANDROID_CAMP(29099): ActivityA started
I/ANDROID_CAMP(29099): ActivityA resumed
I/ANDROID_CAMP(29099): ActivityA paused
I/ANDROID_CAMP(29099): ActivityB created
I/ANDROID_CAMP(29099): ActivityB started
I/ANDROID_CAMP(29099): ActivityB resumed
I/ANDROID_CAMP(29099): ActivityA stopped
I/ANDROID_CAMP(29099): ActivityB paused
I/ANDROID_CAMP(29099): ActivityA result returns ok
```



Let’s try it!

Code Exercise 6



Goal: Implicitly Launch Google Maps

1. Open /android-camp/unit-3/exercise-6
2. Update `local.properties` with SDK path

Notice update to resource files

res/layout/activity_b.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:background="@color/dark_yellow"
              android:padding="8dip">
    <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/lbl_activity_b"
            android:gravity="center_horizontal"
            android:textSize="@dimen/font_large"
            android:textColor="@color/light_yellow"
            android:paddingBottom="16dip"/>

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
                  android:orientation="vertical"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_gravity="center_horizontal">

        <EditText android:id="@+id/location"
                  android:layout_width="300dp"
                  android:hint="Enter a location query"
                  android:layout_height="wrap_content"/>

         <Button android:id="@+id/btn_start_map"
                 android:layout_height="wrap_content"
                 android:layout_width="wrap_content"
                 android:text="@string/btn_start_map"
                 android:onClick="findOnMap"
                 android:layout_gravity="center_horizontal">
        </Button>
    </LinearLayout>

</LinearLayout>
```

Notice update to ActivityB

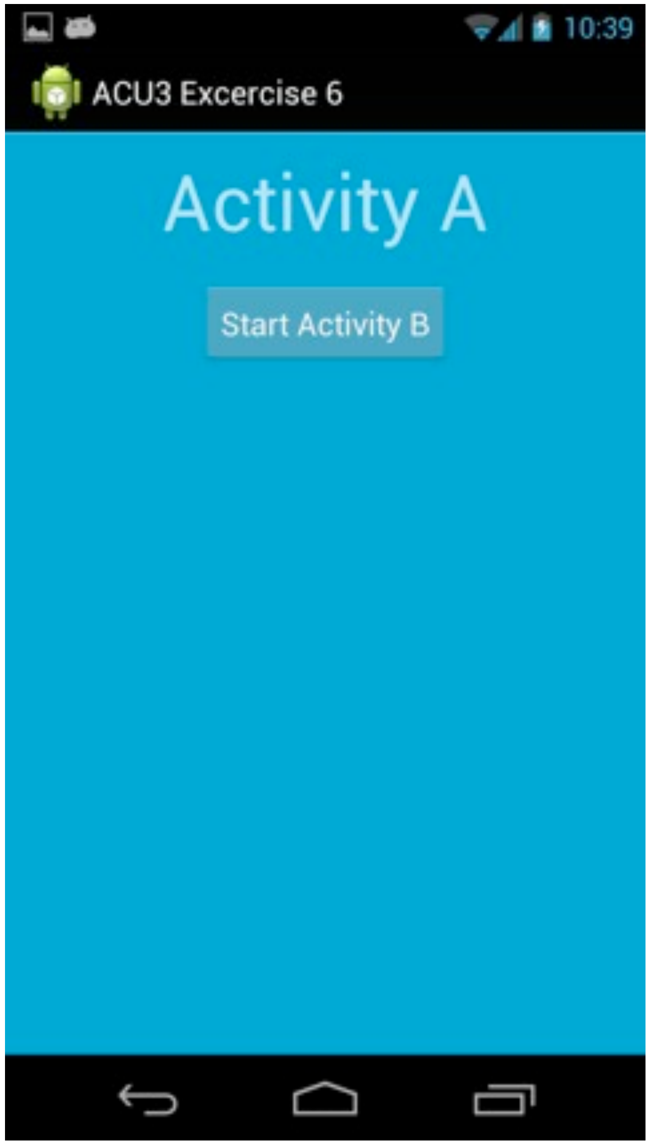
```
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;

public class ActivityB extends Activity {
   ...

   public void findOnMap(View view) {
        String action = Intent.ACTION_VIEW;
        EditText location = (EditText)findViewById(R.id.location);
        Uri uri = Uri.parse("geo:0,0?q=" + location.getText());
        Intent intent = new Intent(action, uri);
        startActivity(intent);
    }

   ...
}
```

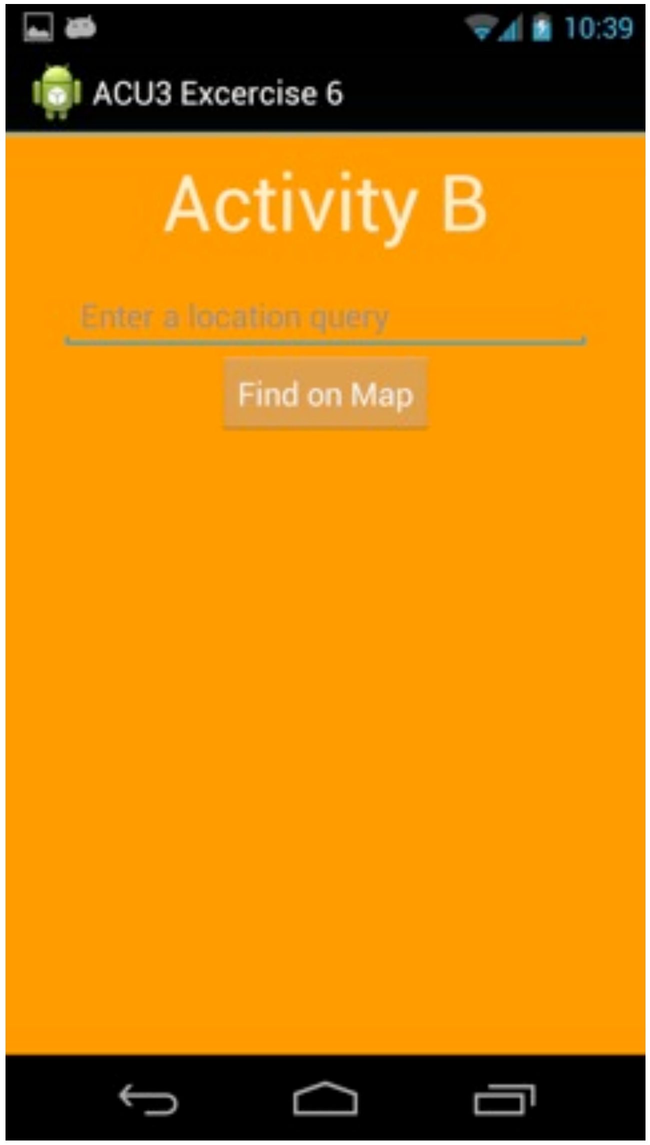
Compile, install and run

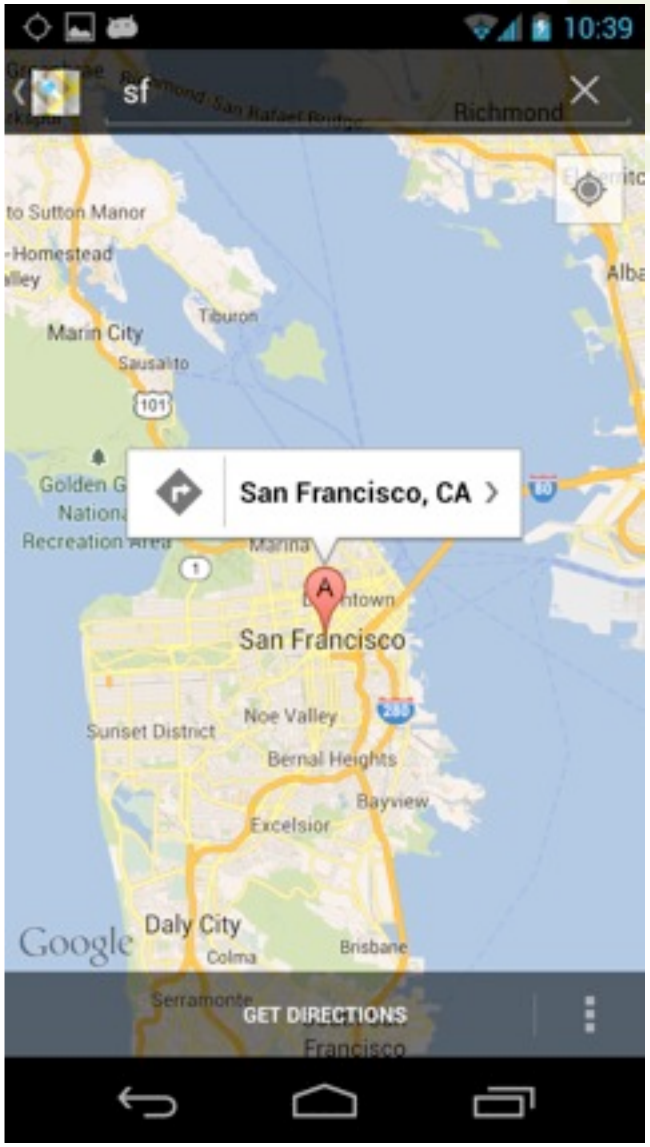
Use the ‘ant’ command

```
brutledge$ ant debug install
Buildfile: /Users/brutledge/workspace/android-camp/build.xml

         ...

install:
     [echo] Installing /Users/brutledge/workspace/android-camp/bin/
ActivityA-debug.apk onto default emulator or device...
     [exec] 784 KB/s (4829 bytes in 0.006s)
     [exec] * daemon not running. starting it now on port 5037 *
     [exec] * daemon started successfully *
     [exec]    pkg: /data/local/tmp/ActivityA-debug.apk
     [exec] Success

BUILD SUCCESSFUL
Total time: 28 seconds
```

Success! Implicit launch of Google Map







The Broadcast Receiver Class

- Extend the abstract class

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class ReceiverA extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i(TAG, “received broadcast intent!”);
    }
}
```

The Android Manifest File

- describe receivers in the manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.google.androidcamp.unit3"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="16" />

    <application android:label="@string/app_name">
        <activity android:name=".ActivityA"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <receiver android:name=".ReceiverA">
            <intent-filter>
                <action android:name="android.intent.action.ACTION_SEND"/>
            </intent-filter>
        </receiver>
</application>
</manifest>
```

Let’s try it!

Code Exercise 7

Goal: Handle Broadcast Intent

1. Open /android-camp/unit-3/exercise-7
2. Update `local.properties` with SDK path

Notice class ReceiverA

```java
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class ReceiverA extends BroadcastReceiver {

    private static final String MSG_POWER_CONNECTED =
        "Power cable has been connected!";
    private static final String MSG_POWER_DISCONNECTED =
        "Power cable has been removed!";

    @Override
    public void onReceive(Context context, Intent intent) {
        // Get the action string from the intent
        String action = intent.getAction();

        // Evaluate the action string and send up a toast message to the screen
        if (action.equals(Intent.ACTION_POWER_CONNECTED)) {
            Toast.makeText(context, MSG_POWER_CONNECTED,
                Toast.LENGTH_SHORT).show();
        } else if (action.equals(Intent.ACTION_POWER_DISCONNECTED)) {
            Toast.makeText(context, MSG_POWER_DISCONNECTED,
                Toast.LENGTH_SHORT).show();
        }
    }
}
```

Notice update to AndroidManifest.xml

Declare `RecevierA` in the manifest

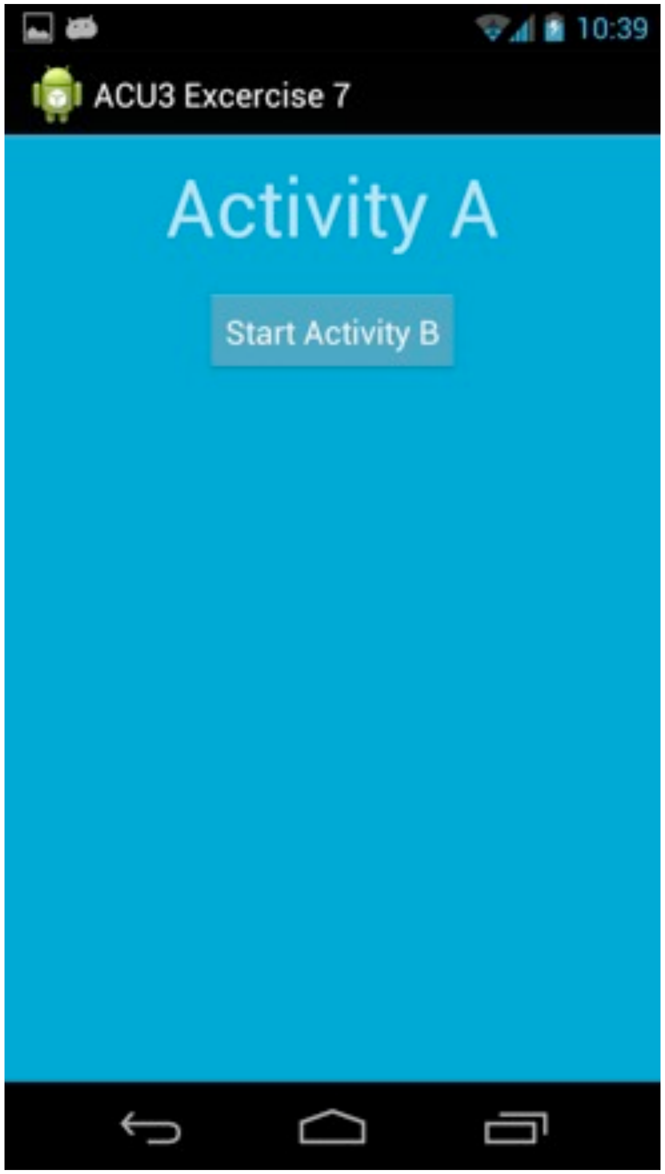
```
<receiver android:name=".ReceiverA">
    <intent-filter>
        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
    </intent-filter>
</receiver>
```

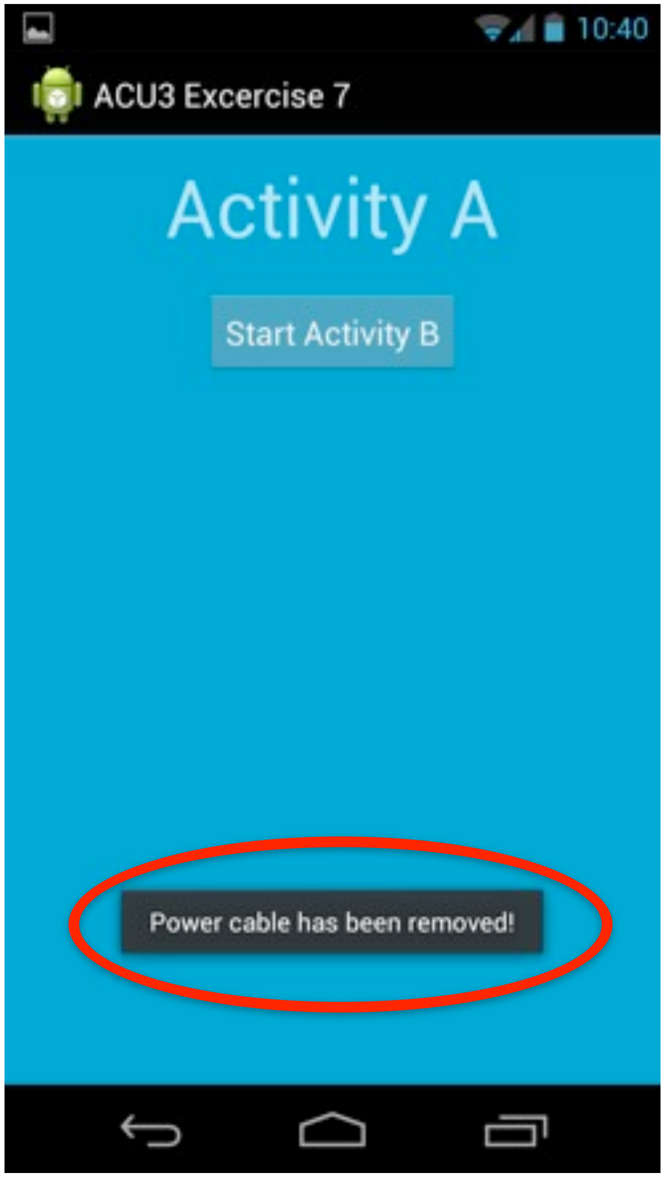
Compile, install and run

Use the ‘ant’ command

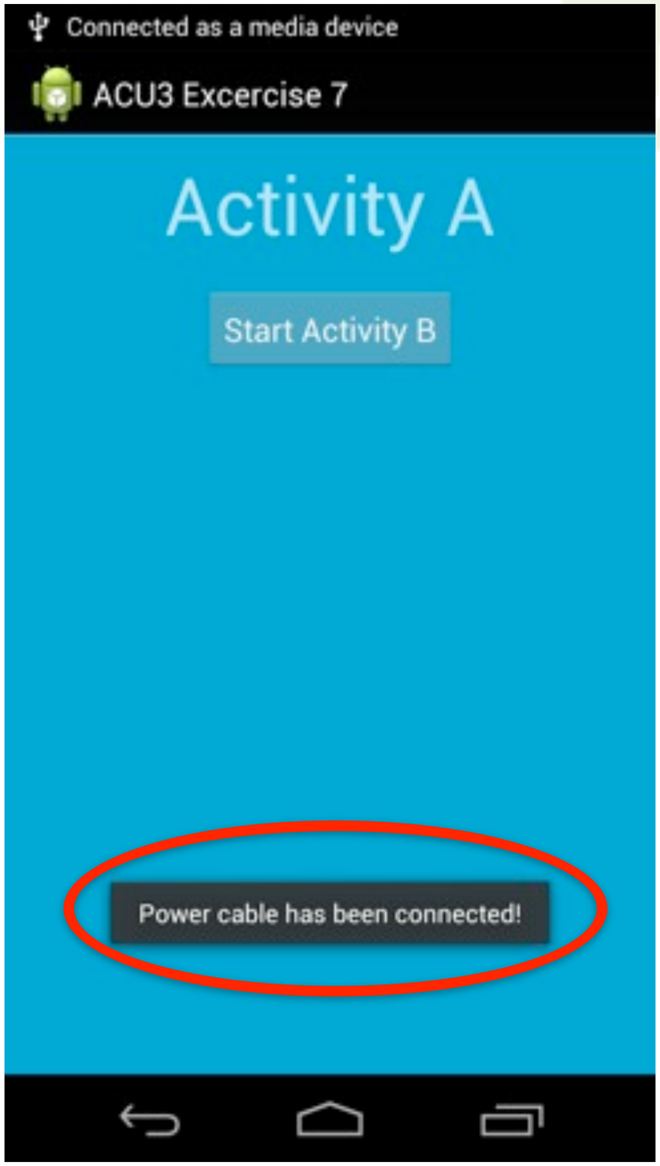
```
brutledge$ ant debug install
Buildfile: /Users/brutledge/workspace/android-camp/build.xml

         ...

install:
     [echo] Installing /Users/brutledge/workspace/android-camp/bin/
ActivityA-debug.apk onto default emulator or device...
     [exec] 784 KB/s (4829 bytes in 0.006s)
     [exec] * daemon not running. starting it now on port 5037 *
     [exec] * daemon started successfully *
     [exec]    pkg: /data/local/tmp/ActivityA-debug.apk
     [exec] Success

BUILD SUCCESSFUL
Total time: 28 seconds
```

Success! Toast when power connects





Power cable has been removed!



Power cable has been connected!

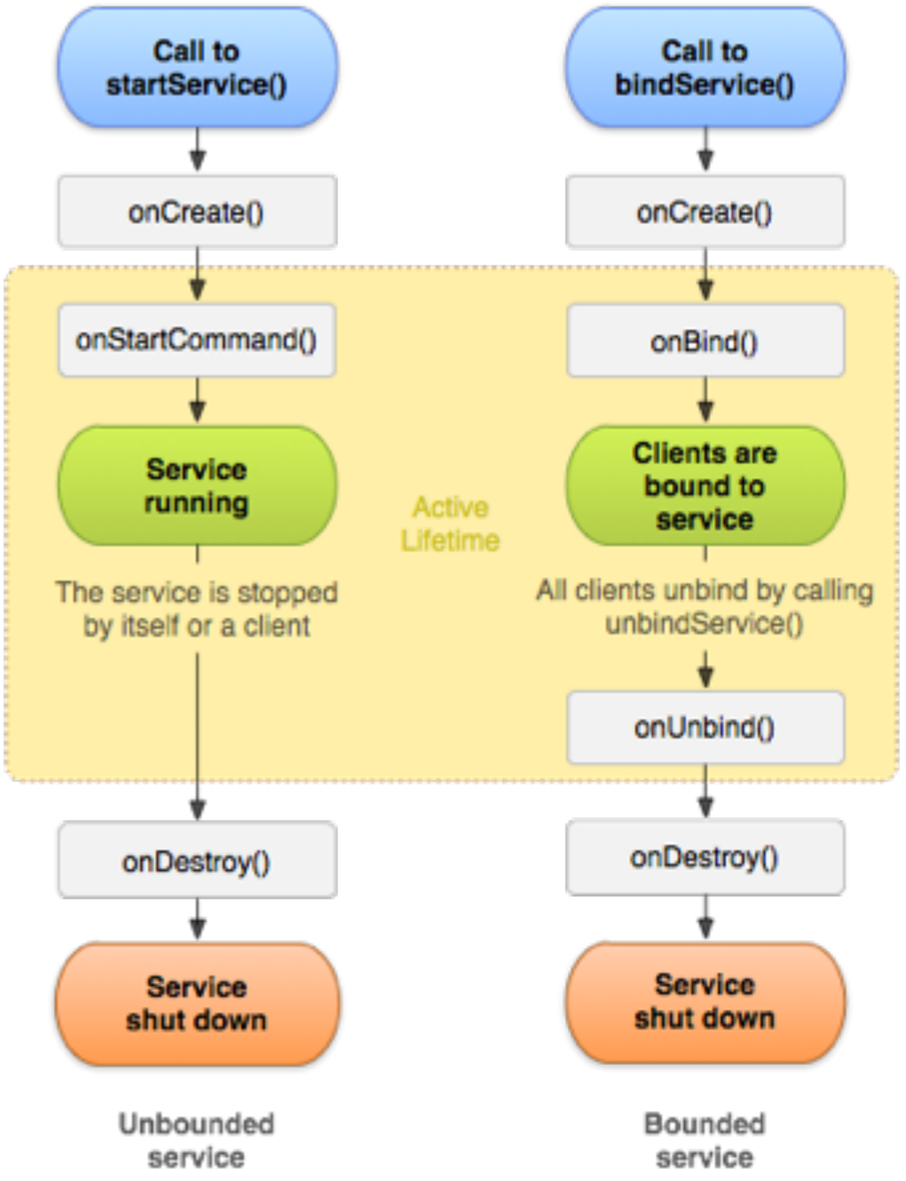
Android Services

Android Services

- Have a background lifecycle
- Are a registered component
- Have two forms
  - "Started"
    - Started by Activity, can outlive it.
    - Should stop itself when its task is done.
  - "Bound"
    - Offers client/server interface.
    - Apps in different processes can interact with it.
    - Is destroyed when **no components are bound to it**

The Service Lifecycle

- Like Activities, Services have a Lifecycle.



IntentService

- IntentService handles a lot “out of the box”
  - Creates default worker thread
  - Creates queue that handles incoming intents
  - Passes them one at a time to onHandleIntent()
  - Stops the service after all requests handled.
- Makes things a lot easier.

What IntentService Looks Like

- Extending is easy. Just implement onHandleIntent()

```
public class HelloIntentService extends IntentService {
    public HelloIntentService() {
        super("HelloIntentService");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // Do neat things!
    }
}
```

- Can also override onCreate(), onStartCommand(), onDestroy()

Let’s try it!

Code Exercise 7

Goal: Start a Service

1. Open /android-camp/unit-3/exercise-8
2. Update `local.properties` with SDK path
3. Update the service to wait 5 seconds, then open a browser to the specified URL

Implement onServiceHandler()

HelloIntentService.java

```
public class HelloIntentService extends IntentService {
        public HelloIntentService() {
            super("HelloIntentService");
        }

        @Override
        protected void onHandleIntent(Intent paramIntent) {
            long endTime = System.currentTimeMillis() + 3*1000;
            while (System.currentTimeMillis() < endTime) {
                synchronized (this) {
                    try {
                        wait(endTime - System.currentTimeMillis());
                    } catch (Exception e) {
                    }
                }
            }
              Uri uri = paramIntent.getData();
              Log.d("HelloIntentService", uri.toString());

              Intent intent = new Intent(Intent.ACTION_VIEW, uri);
              intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
              startActivity(intent);
        }
}
```

Add Service to the Manifest

```
   <application
...
     <service
       android:exported="false"
       android:name=".HelloIntentService" />
```

Create the button in XML

res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent" >

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      android:layout_centerVertical="true"
      android:text="@string/hello_world"
      tools:context=".MainActivity" />

    <Button android:id="@+id/btn_start_service"
          android:layout_height="wrap_content"
          android:layout_width="wrap_content"
          android:text="Activate"
          android:onClick="onClick"
          android:layout_gravity="center_horizontal" />

</RelativeLayout>
```

Start the service on button click

res/layout/activity_main.xml

```
public void onClick(View view) {
    Intent intent = new Intent(this, HelloIntentService.class);
    Uri uri = Uri.parse("http://www.google.com");
    intent.setData(uri);
    startService(intent);
}
```

That’s All For Unit 3