

An abstract, high-speed photograph of a water splash or droplet impact, rendered in a light, ethereal style. The splash is composed of various shades of grey and white, with fine droplets and larger, more turbulent sections of water. It curves from the top left towards the right side of the frame, creating a sense of motion and fluidity.

# Week 14

## @wesreisz

# Agenda



- Review Test
- Android Services
- Android Notifications
- Not My Music Updates



# Review Test

Quiz 2 - Results ☆

File Edit View Insert Format Data Tools Add-ons Help Comments Share

fx | Long Answer

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	question																		Total	Of	
2	Fill in the Blank																				
3	1																1		1	/17	
4	2			1															1	/17	
5	3				1		1			1	1	1	1	1	1	1			8	/17	
6	4	1		1	1				1		1	1			1	1			8	/17	
7	5																		0	/17	
8	6	1	1						1		1	1				1	1		7	/17	
9	7			1					1	1		1	1				1		6	/17	
10	8	1				1		1	1	1	1	1	1	1	1	1	1		11	/17	
11	9	1		1	1	1	1	1			1	1	1	1	1	1	1		13	/17	
12	10												1						1	/17	
13	11	1	1	1	1		1	1	1	1		1	1			1	1	1	13	/17	
14	12	1				1									1		1		4	/17	
15	13			1	1		1	1	1	1	1	1	1	1			1	1	12	/17	
16	14																		0	/17	
17	15		1			1				1	1	1			1				6	/17	
18	Code Reading																				



<http://bit.ly/1IcgfP9>

# Android Basics

## App Components

Intents and Intent Filters

Activities

Services

Content Providers

App Widgets

Processes and Threads

<http://developer.android.com/guide/components/index.html>



# Android Service

A [Service](#) is an application component that can perform long-running operations in the background and does not provide a user interface. Another application component can start a service and it will continue to run in the background even if the user switches to another application.

Additionally, a component can bind to a service to interact with it and even perform interprocess communication (IPC).





# Service

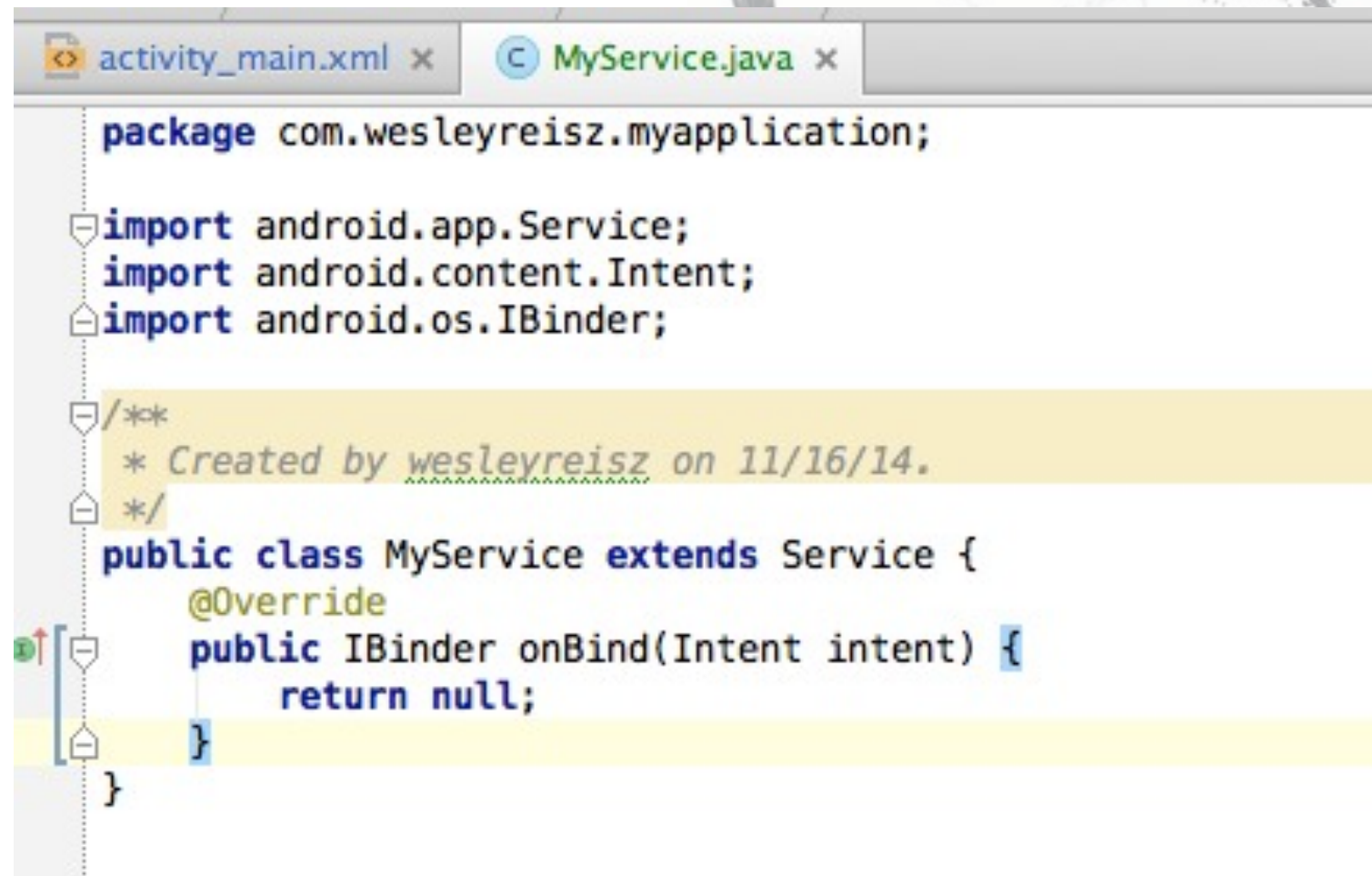


# Android Service: The Basics



- To create a service, you must create a subclass of [Service](#) (or one of its existing subclasses)

# Example



```
activity_main.xml x  MyService.java x
package com.wesleyreisz.myapplication;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;

/**
 * Created by wesleyreisz on 11/16/14.
 */
public class MyService extends Service {
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```



# OnBind



- [onBind\(\)](#)
- The system calls this method when another component wants to bind with the service (such as to perform RPC), by calling [bindService\(\)](#). In your implementation of this method, you must provide an interface that clients use to communicate with the service, by returning an [IBinder](#). You must always implement this method, but if you don't want to allow binding, then you should return null.



# Important Methods

- [startService\(\)](#). Once this method executes, the service is started and can run in the background indefinitely. If you implement this, it is your responsibility to stop the service when its work is done, by calling [stopSelf\(\)](#) or [stopService\(\)](#). (If you only want to provide binding, you don't need to implement this method.)
- [onCreate\(\)](#)  
The system calls this method when the service is first created, to perform one-time setup procedures (before it calls either [onStartCommand\(\)](#) or [onBind\(\)](#)). If the service is already running, this method is not called.
- [onDestroy\(\)](#)  
The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc. This is the last call the service receives.

# Manifest

```
<manifest ... >  
  
  <application ... >  
  
    <service android:name=".ExampleService" />  
  
  </application>  
  
</manifest>
```



# Demo



153ef72 added thread

093a1e0 blocked ui thread

2588168 example of workign service

4c406ba added presentation and added a service template

4c60786 initial project with Application Class



# Notifications

A **notification** is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.





4:23 PM

Thursday, October 2



### The Big Meeting

4:15 – 5:15 PM

Big Conference Room



MAP



EMAIL GUES...



### New Google+ notifications

3:44 PM

Earl Liibyrd: Added you back

2



### Screenshot captured.

4:22 PM

Touch to view your screenshot.



### Keep photos & videos backed..

4:22 PM

Touch to get free private storage on Google



### 3 new messages

4:11 PM

kumlata.r.swankatranami@gmail.com

3



### 3 new messages

2:42 PM

(754) 263-8267, 456

3 cards

WESLEY  
REISZ



# Notifications

- You specify the UI information and actions for a notification in a **NotificationCompat.Builder** object. To create the notification itself, you call **NotificationCompat.Builder.build()**, which returns a Notification object containing your specifications. To issue the notification, you pass the Notification object to the system by calling **NotificationManager.notify()**.



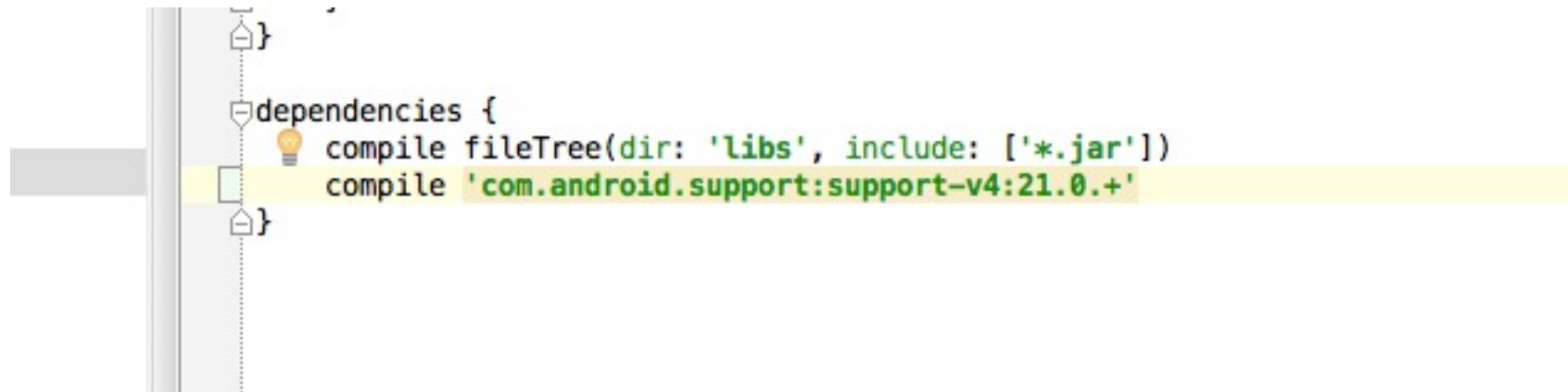
# Creating a Notification

- A Notification object must contain the following:
  - A small icon, set by `setSmallIcon()`
  - A title, set by `setContentTitle()`
  - Detail text, set by `setContentText()`



# NOTE:

- Add the support libraries or Android Studio will complain:





# Simple Notification

```
//build up your notification. btw, this is a builder pattern
NotificationCompat.Builder mBuilder =
    new NotificationCompat.Builder(getApplicationContext());

mBuilder.setSmallIcon(android.R.drawable.ic_dialog_alert);
mBuilder.setContentTitle("Notification " + count);
mBuilder.setContentText(" " + count);

//send it
notificationManager.notify(
    count,
    mBuilder.build()
);
}
```



# Demo



8260f7f added pending intent

53307f2 passed a message into service and on to notification

701bb98 simple notification



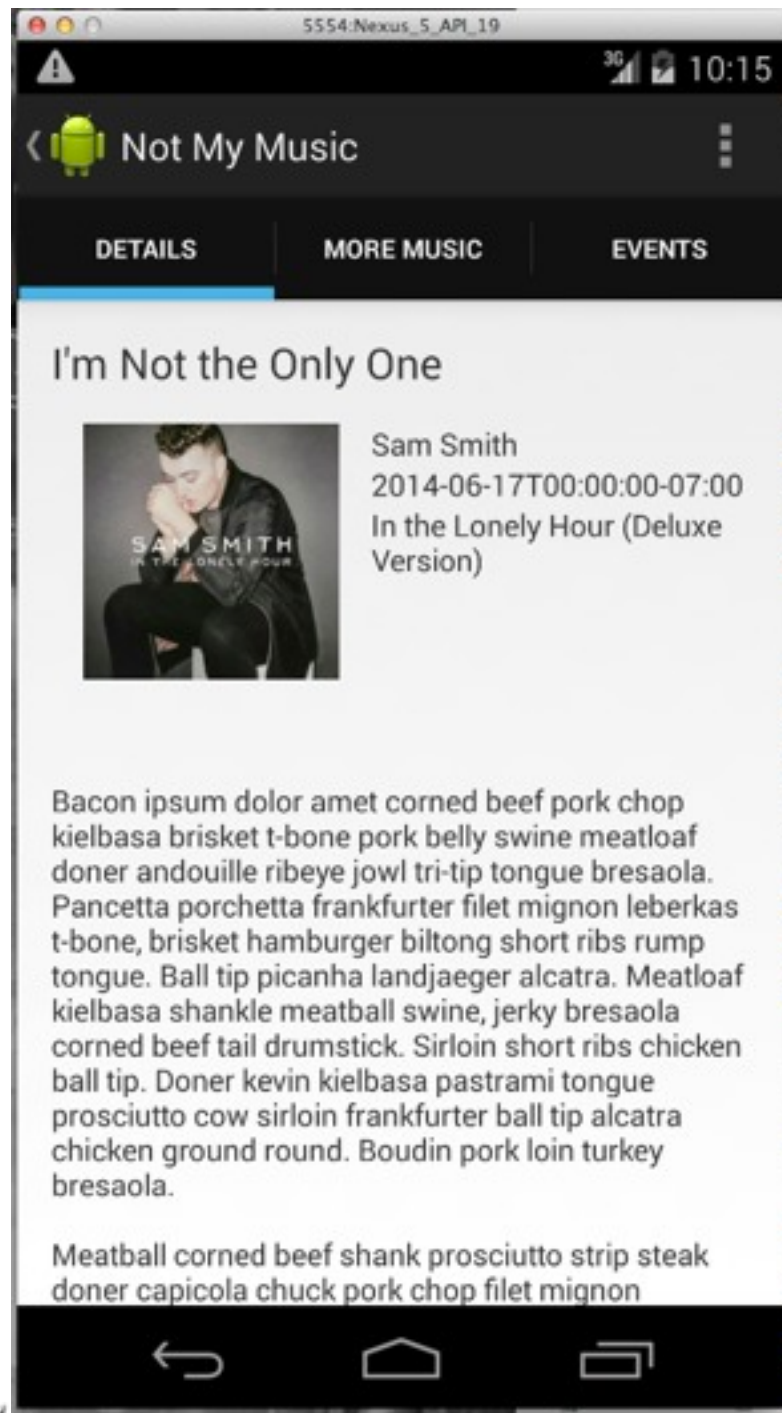
# Agenda



- Review Test
- Android Services
- Android Notifications
- Not My Music Updates



# Not My Music Updates





```

    */
    public class MyTabListener implements ActionBar.TabListener{
        Activity mActivity;
        private Fragment fragment;
        FrameLayout fm;

        public MyTabListener(Activity activity, Fragment fragment){
            this.fragment = fragment;
            this.mActivity = activity;
        }
        private Fragment moreFragment;

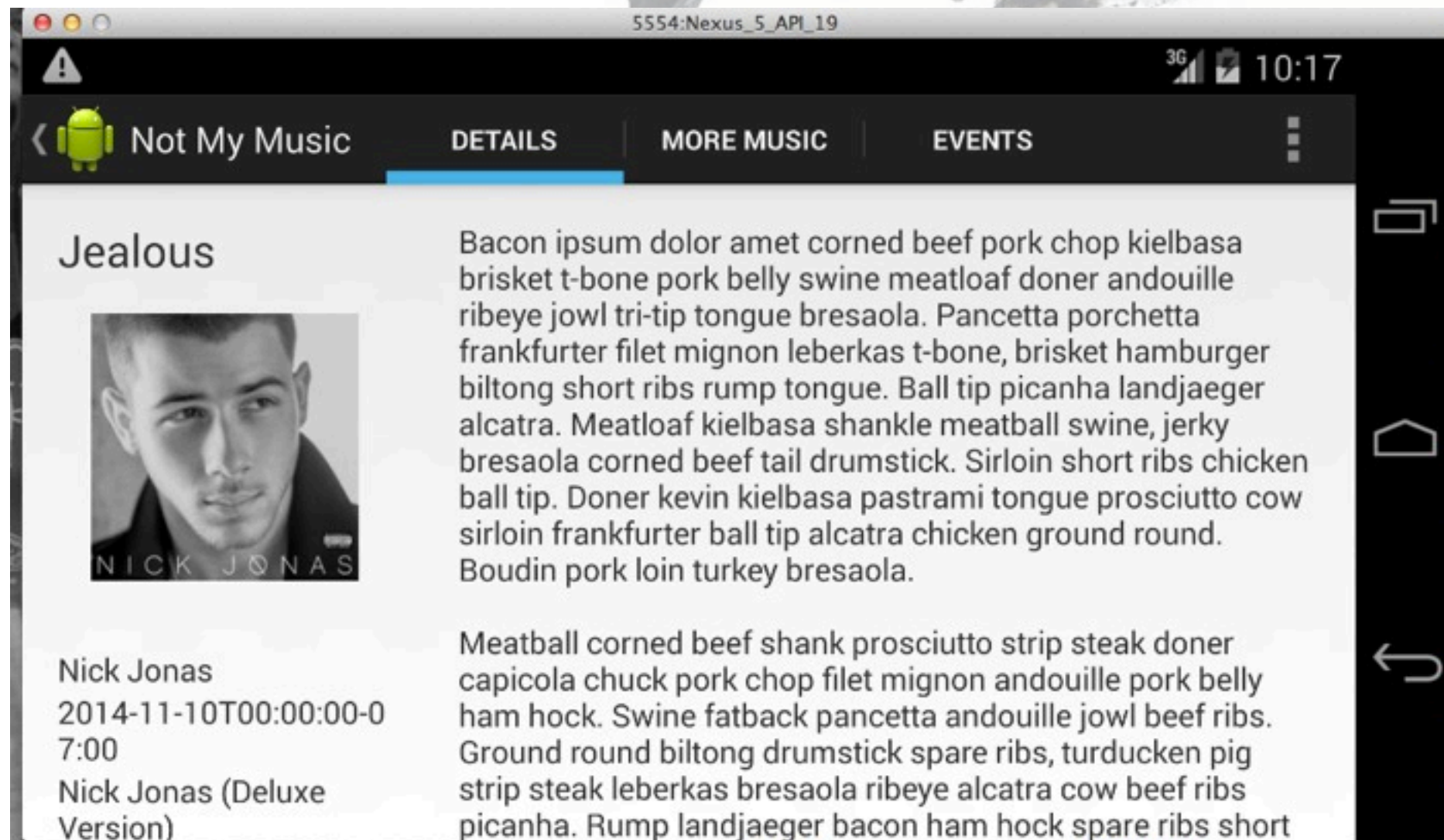
        @Override
        public void onTabSelected(ActionBar.Tab tab, FragmentTransaction ft) {
            ft.replace(R.id.container, fragment);
            if(fragment instanceof DetailsFragment){
                Log.d("TEST", "Loading detailsfragment");
                fm = (FrameLayout)mActivity.findViewById(R.id.containerMore);
                fm.setVisibility(View.VISIBLE);
                moreFragment = new MoreSongsFragment();
                ft.replace(R.id.containerMore, moreFragment);
            }
        }

        @Override
        public void onTabUnselected(ActionBar.Tab tab, FragmentTransaction ft) {
            ft.remove(fragment);
            if(moreFragment!=null && fm !=null){
                fm.setVisibility(View.GONE);
                ft.remove(moreFragment);
            }
        }


        @Override
        public void onTabReselected(ActionBar.Tab tab, FragmentTransaction ft) {
        }
    }

```

# Not My Music Updates





- 
- ▶ drawable-xxhdpi
  - ▶ layout
  - ▼ layout-land
    - ▶ activity\_song\_details.xml
    - ▶ fragment\_details.xml
  - ▼ menu
    - ▶ menu\_details.xml
    - ▶ menu\_song\_grid.xml
  - ▶ values

# Agenda



- Review Test
- Android Services
- Android Notifications
- Not My Music Updates

