CS 646 Android Mobile Application Development
Spring Semester, 2015
Doc 19 Broadcasts, Services, Notifications
Apr 16, 2015

Copyright ©, All rights reserved. 2015 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/openpub/) license defines the copyright on this document.

Big Nerd Ranch Chapters

Chapter 29 Background Services
Chapter 30 Broadcast Intents

Java Nested Classes

```
class OuterClass {
    ...
    class InnerClass {
        ...
    }
}
```

Inner Class

```
class OuterClass {
    ...
    static class StaticNestedClass {
        ...
    }
    class InnerClass {
        ...
    }
}
```

Static Nested Class

Inner Class

```
class OuterClass {
    ...
    class InnerClass {
        ...
    }
}
```

```
OuterClass a = new OuterClass();

OuterClass.InnerClass b = a.new InnerClass();
```

Need instance of OuterClass to create instance of InnerClass

OuterClass Creating Instance of InnerClass

```
public class OuterClass {

    public void example() {
        InnerClass c = new InnerClass();
    }

    class InnerClass {

    }

}
```

InnerClass Accessing OuterClass Fields

```
public class OuterClass {
    int x = 0;
    int y = -1;
    class InnerClass {
        int y = 2;

        public void sample() {
            int a = OuterClass.this.x;  //0
            int b = x;                  //0
            int c = OuterClass.this.y;  //-1
            int d = y;                  //2
        }
    }
}
```

Broadcasts

Broadcast Intents

Intents to send information about an event to multiple components

OS level broadcasts

App broadcasts
- To other apps
- To compontents in the same app

Some OS broadcast intents

ACTION_POWER_DISCONNECTED
AIRPLANE_MODE
BATTERY_LOW
BOOT_COMPLETED
CAMERA_BUTTON
DATA_SMS_RECEIVED
DOWNLOAD_COMPLETE
DREAMING_STARTED
HEADSET_PLUG
NEW_OUTGOING_CALL
NEW_PICTURE
NEW_VOICEMAIL
NETWORK_IDS_CHANGED
REBOOT
RSSI_CHANGED
SCREEN_ON
WIFI_STATE_CHANGED

Full list at your Android installation

android-sdks\platforms\android-xx\data
\broadcast_actions

Receiving Broadcasts

- Register for broadcasts
  - In Manifest
  - Dynamically in code
- Subclass BroadcastReceiver
  - Implement onReceive(Context context, Intent intent)

Broadcasts When App is not running

App does not need to be running to receive broadcast

App only runs onReceive method

Start Service if need to perform long run operation

Example - Notification of Device starting

```
public class StartupReceiver extends BroadcastReceiver {
    private static final String TAG = "StartupReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i(TAG, "Received broadcast intent: " + intent.getAction());
    }
}
```

Example - Notification of Device starting

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.bignerdranch.android.photogallery"
  android:versionCode="1"
  android:versionName="1.0" >
  <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

  <application>
    <activity></activity>
      <receiver android:name=".StartupReceiver">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
    </receiver>
  </application>

</manifest>
```

Dynamically Register for BroadCast

Create IntentFilter with Broadcast action

Register the filter with broadcast receiver

Dynamically Register for BroadCast

In activity

```
public static final String ACTION_SHOW_NOTIFICATION =
        "com.bignerdranch.android.photogallery.SHOW_NOTIFICATION";
```

```
private BroadcastReceiver mOnShowNotification = new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            Toast.makeText(getActivity(),
                    "Got a broadcast:" + intent.getAction(),
                    Toast.LENGTH_LONG)
                .show();
        }
  };
```

Dynamically Register for BroadCasr

```
public void onResume() {
      super.onResume();
      IntentFilter filter = new IntentFilter(ACTION_SHOW_NOTIFICATION);
      registerReceiver(mOnShowNotification, filter);
  }


  public void onPause() {
     super.onPause();
     getActivity().unregisterReceiver(mOnShowNotification);
  }
```

Sending a Broadcast

Create an intent with Broadcast action

Use Activity sendBroadcast(Intent) method

Dont forget that intents can hold data

Sending Data back to Broadcast Orginator

Use ordered broadcasts

```
sendOrderedBroadcast(
    Intent intent,
    String receiverPermission,
    BroadcastReceiver resultReceiver, //receives results
    Handler scheduler,
    int initialCode,
    String initialData,
    Bundle initialExtras)
```

In receiver return data via

```
setResult(int,String,Bundle)
```

Services

Service

Runs in the background

No user interaction

Runs indefinitely

Runs in thread of hosting process

- Create new thread to do work

Started Service

Usually used to perform single operation in background

    Download a file

Normally does not return return result to caller

    Can do so using broadcast or Messengers

When operation is done service should stop itself

Started by sending the following to the context(Activity)

    startService (Intent service)

Bound Service

App component binds to the service

Service has client-server interface

App component can call methods on service, get responses

Service is destroyed when last client unbinds

Bind to a service by sending to the context:

```
bindService (Intent service, ServiceConnection conn, int flags)
```

Started & Bound

Service can be both

But must start as started service

Does not end when last client unbinds

Basics

Service runs on main thread so you likely need to create separate thread

Main Methods

- onStartCommand()
- onBind()
- onCreate()
- onDestroy()

Service Lifecycle - Started Service

Started when Context.startService(Intent) is called

Then service's methods are called
- onCreate()
- onStartCommand(Intent intent,int flags, int startId)

Service runs until one of following is called
- Context.stopService(Intent)
- stopSelf()

Only one of a service runs

Implement onDestroy() to clean up

Process Lifecycle

Services are killed if low on memory

- If currently executing onCreate(), onStart(), or onDestroy()
  - service not killed
- If service has been started
  - process is less important currently visible processes
  - process is more important than processes not visible & don't have service
- If clients are bound to the service
  - process is as important as the most important client

Permissions

Need to declare in manifest

```
<service android:name=".AvitarService" />
```

Other applications need `<uses-permission>` to use service

Started Service

- Subclass
  - Service
    - General Service
  - IntentService
    - Does most of the work for you
    - Create thread to do work in
    - But does not handle multiple request simultaneously
    - Queues requests up and handles them one at a time
    - Just implement onHandleIntent() method

Bound Service

Client must implement the ServiceConnection interface

    onServiceConnected(ComponentName name, IBinder service)
    onServiceDisconnected(ComponentName name)

Client calls bindService() which calls service's onBind()

onBind returns an IBinder object

Client uses IBinder object to send messages to service

Client sends following to context to unbind

    unbindService (ServiceConnection conn)

onBind

Called only once after service is created

System stores the IBinder object

When second client calls bindService() stored IBinder object is sent to client

IBinder - the easy way

This only works if Service and Client are in same application

Subclass Binder

Add public method getService() which returns your service

Return the Binder subclass object in service's onBind() method

Client gets binder object in onServiceConnected

Client casts binder object to service type

IBinder Service Example

```
public class LocalService extends Service {
    private final IBinder mBinder = new LocalBinder();
    private final Random mGenerator = new Random();

    public IBinder onBind(Intent intent) {
        return mBinder;
    }

    public int getRandomNumber() {
      return mGenerator.nextInt(100);
    }

     public class LocalBinder extends Binder {
        LocalService getService() {
            return LocalService.this;
        }
    }
}
```

Using the Service

```
public class BindingActivity extends Activity implements ServiceConnection{
      LocalService mService;
      boolean mBound = false;

      public void onServiceConnected(ComponentName className,IBinder service) {
          LocalBinder binder = (LocalBinder) service;
          mService = binder.getService();
          mBound = true;
       }

       @Override
       public void onServiceDisconnected(ComponentName arg0) {
          mBound = false;
       }
```

```
protected void onStart() {
    super.onStart();
    Intent intent = new Intent(this, LocalService.class);
    bindService(intent, this, Context.BIND_AUTO_CREATE);
}

@Override
protected void onStop() {
    super.onStop();
    if (mBound) {
        unbindService(mConnection);
        mBound = false;
    }
}
```

Calling the Service

```
public void onButtonClick(View v) {
    if (mBound) {
        int num = mService.getRandomNumber();
        Toast.makeText(this, "number: " + num, Toast.LENGTH_SHORT).show();
    }
}
```

How It works

```
Intent intent = new Intent(this, LocalService.class);
bindService(intent, this, Context.BIND_AUTO_CREATE);
```

Creates Service
On Service Calls

```
public IBinder onBind(Intent intent) {
      return mBinder;
  }
```

Result is passed to Activity via

```
public void onServiceConnected(ComponentName className,IBinder service)
```

How It works

```
public void onServiceConnected(ComponentName className,IBinder service) {
        LocalBinder binder = (LocalBinder) service;
        mService = binder.getService();
        mBound = true;
}
```

Now Activity has reference to service

IBinder - Messenger

Works across applications

Handles request one at time

Queues requests up if needed

Service using Messenger

```
public class MessengerService extends Service {
    static final int MSG_SAY_HELLO = 1;

     class IncomingHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {
            switch (msg.what) {
                case MSG_SAY_HELLO:
                    Toast.makeText(getApplicationContext(), "hello!",
                                        Toast.LENGTH_SHORT).show();
                    break;
                default:
                    super.handleMessage(msg);
            }
        }
    }
```

Example from:http://developer.android.com/guide/topics/fundamentals/bound-services.html

Service using Messenger

```
final Messenger mMessenger = new Messenger(new IncomingHandler());

     @Override
    public IBinder onBind(Intent intent) {
        return mMessenger.getBinder();
    }
}
```

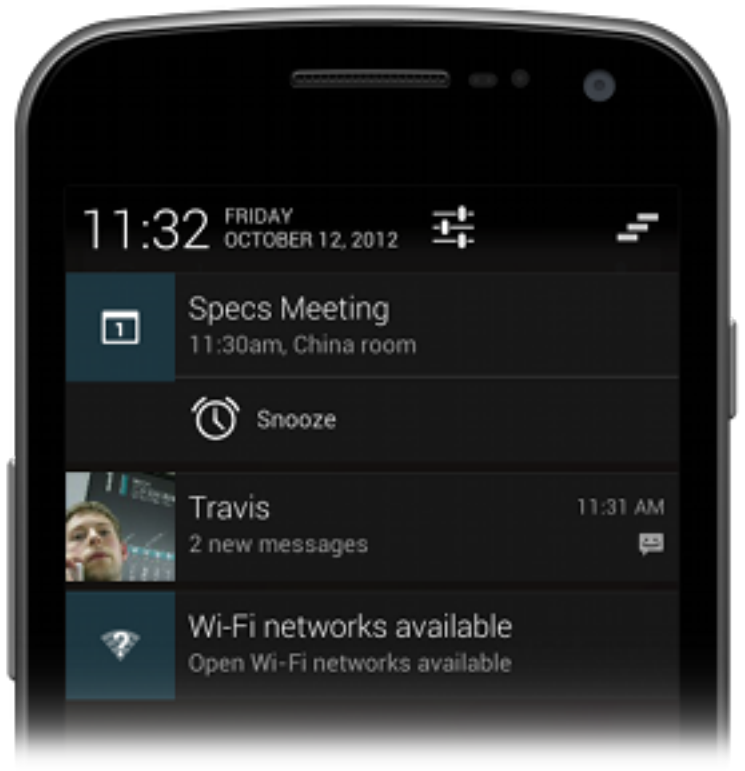
Notifications

Notifications

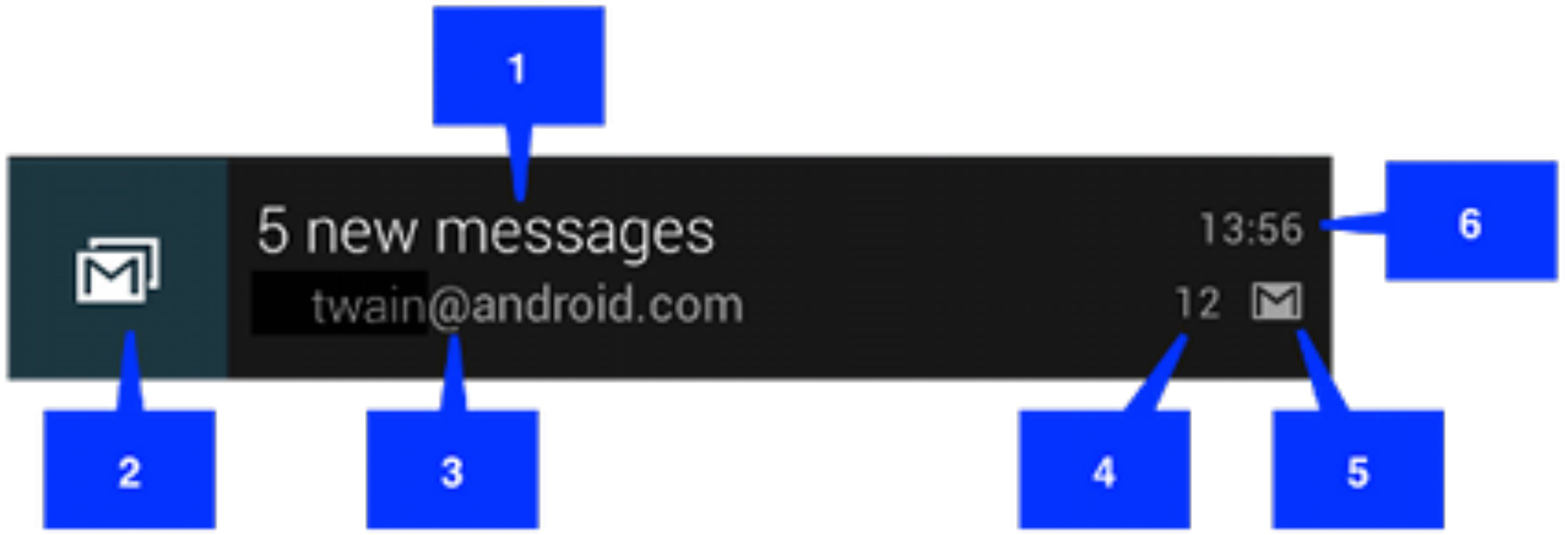
Normal View

Big View

Added in Android 4.1



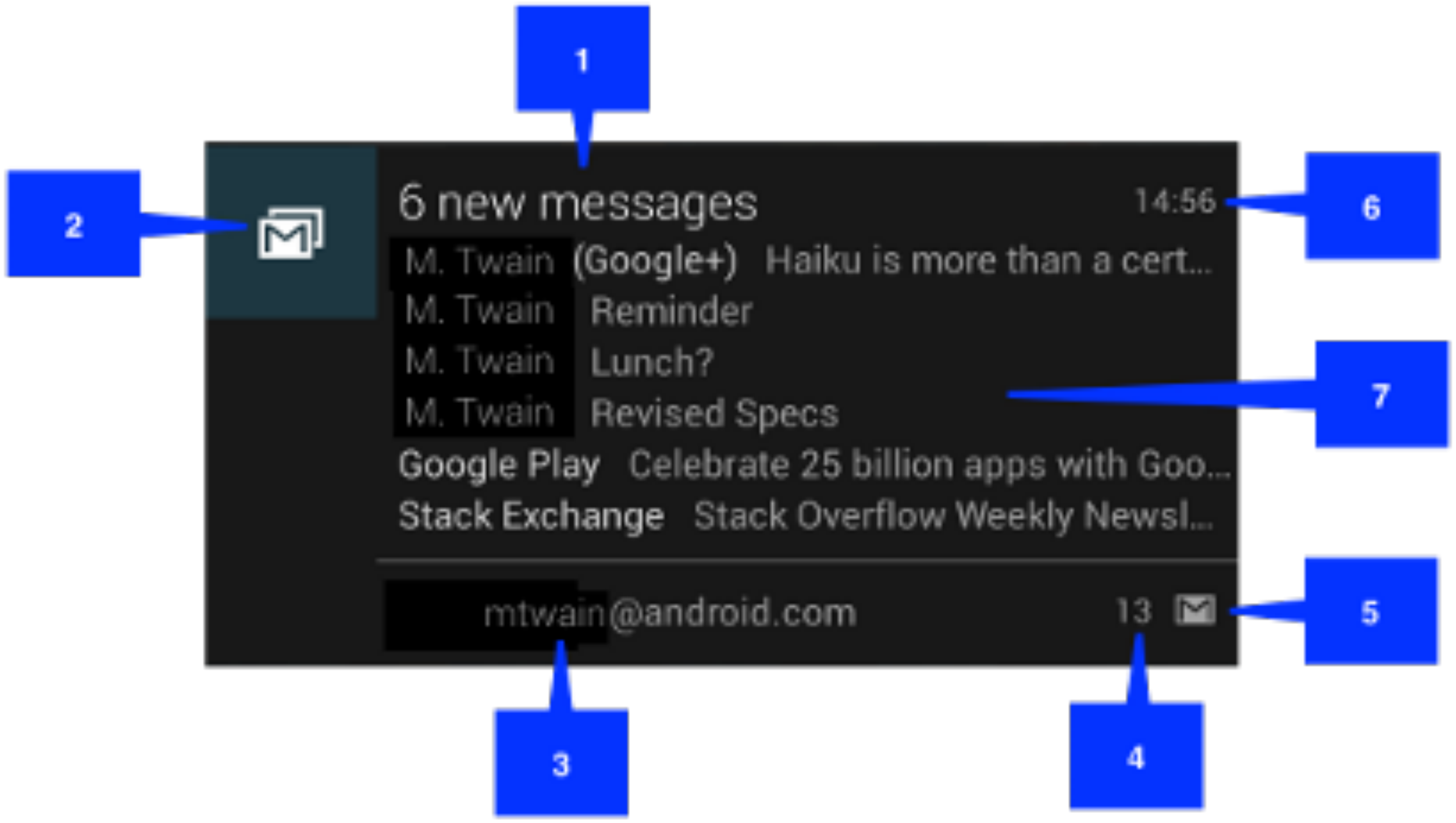
Normal View



5 new messages 13:56
[redacted] twain@android.com 12

1 Content Title
2 Large icon
3 Content text
4 Content info
5 Small icon
6 Time notification was issued

Big View


6 new messages 14:56
M. Twain (Google+) Haiku is more than a cert...
M. Twain Reminder
M. Twain Lunch?
M. Twain Revised Specs
Google Play Celebrate 25 billion apps with Goo...
Stack Exchange Stack Overflow Weekly Newsl...
mtwain@android.com 13

1 Content Title
2 Large icon
3 Content text
4 Content info
5 Small icon
6 Time notification was issued
7 Detail area

Creating a Notification

Build by Hand

Notification Builder

- Makes it easy to create notifications
- Added in Android 3.0
- A version is in Support library

Notification.Builder	NotificationCompat.Builder
Android 3.0+	In Support Library

Creating a Notification

Notification must have

- Small icon
- Title
- Detailed text

Notification should have

- At least one action (Intent)
    - Launches Activity in app when notification is selected

Notifications.Builder

```
Notification note = new Notification.Builder(mContext)
        .setContentTitle("New mail from " + sender.toString())
        .setContentText(subject)
        .setSmallIcon(R.drawable.new_mail)
        .setLargeIcon(aBitmap)
        .build();
```

Notifications & Back Stack

Notifications often jump into middle of an App

- Example - Gmail notification
  - Leads to Activity that shows email message
  - Not the first activity when starting Gmail app

Back button

- Should lead to the logical "previous" activity in the app

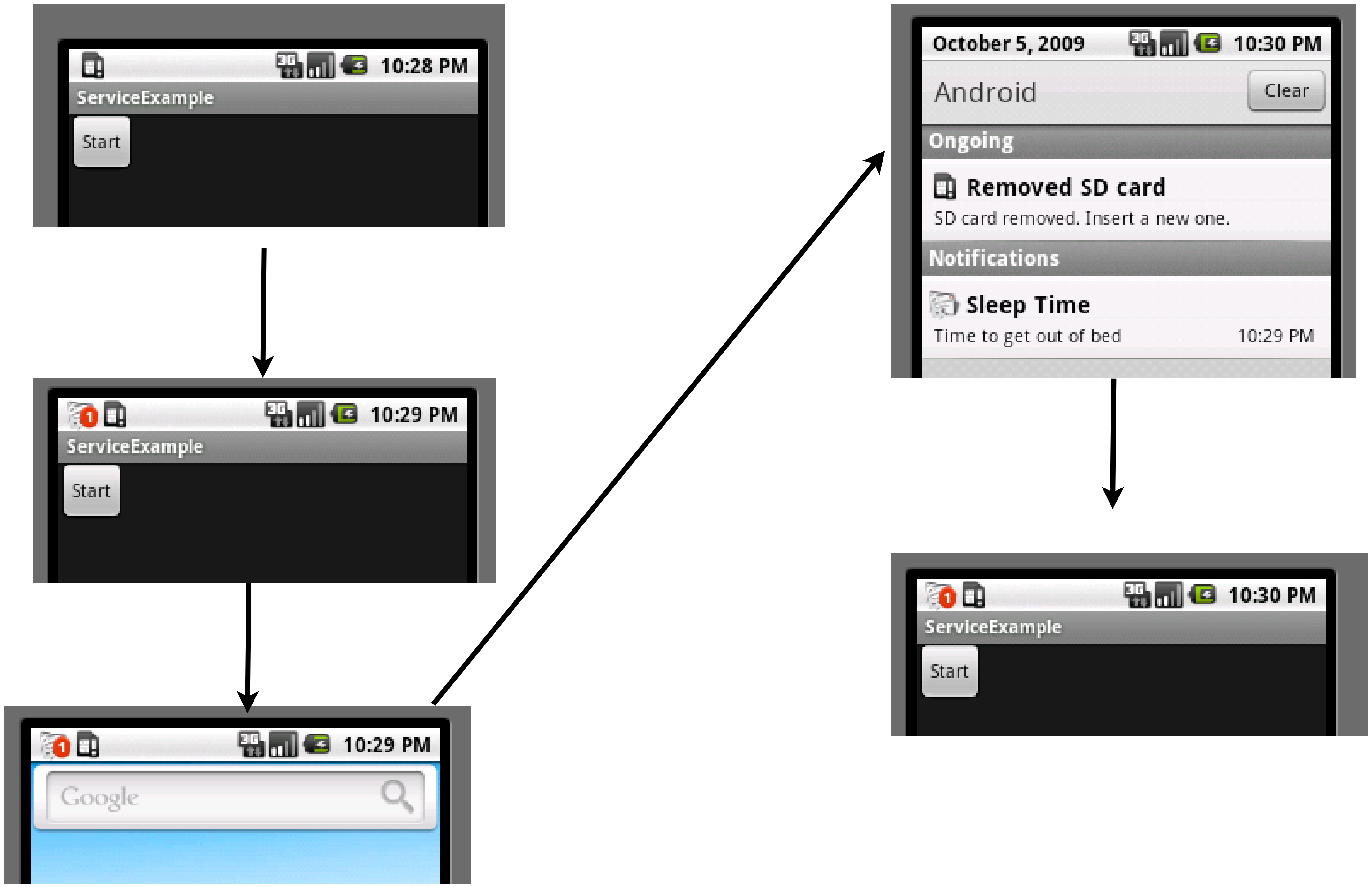
TaskStackBuilder

- Use to build the Back Stack

Example

```
Intent resultIntent = new Intent(this, ResultActivity.class);
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);

// Adds the back stack
stackBuilder.addParentStack(ResultActivity.class);

// Adds the Intent to the top of the stack
stackBuilder.addNextIntent(resultIntent);

// Gets a PendingIntent containing the entire back stack
PendingIntent resultPendingIntent =
        stackBuilder.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);

NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
builder.setContentIntent(resultPendingIntent);
NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(id, builder.build());
```

Notification



Removed SD card
SD card removed. Insert a new one.

Sleep Time
Time to get out of bed 10:29 PM

Send Notification on Button Click

```
public class ServiceExample extends Activity implements View.OnClickListener {
    private static final int NOTIFY_ID = 1123;

    private int count = 0;

    public void onClick(View v) {
        TimerTask task = new TimerTask() {
            public void run() {
                sendNotification();
            }
        };
        new Timer().schedule(task, 5000);
    }

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button ok = (Button) findViewById(R.id.start);
        ok.setOnClickListener(this);
    }
```

The Notification

```
    private void sendNotification() {
        Notification note = new Notification(R.drawable.icon, "Wake Up!",
                System.currentTimeMillis());
        PendingIntent intentToStart = PendingIntent.getActivity(this, 0,
                new Intent(this, ServiceExample.class),
                Intent.FLAG_ACTIVITY_NEW_TASK);

        note.setLatestEventInfo(this, "Sleep Time", "Time to get out of bed",
                intentToStart);
        note.number = ++count;
        note.defaults = Notification.DEFAULT_VIBRATE;
        NotificationManager manager = (NotificationManager)
                                        getSystemService(NOTIFICATION_SERVIC
        manager.notify(NOTIFY_ID, note);
    }
}
```

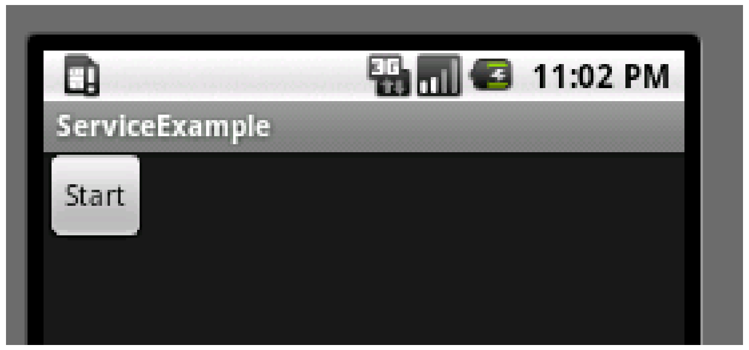
Need permission to Vibrate

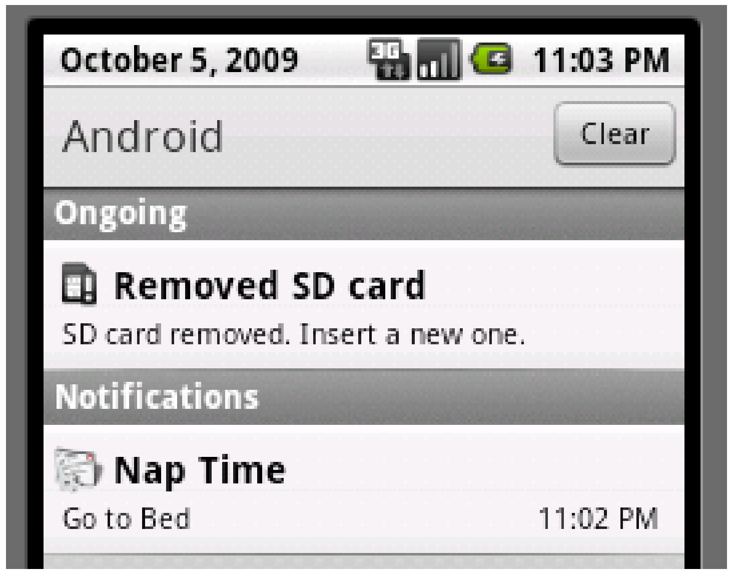
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="edu.sdsu.cs.whitney"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ServiceExample"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
          </application>
    <uses-sdk android:minSdkVersion="3" />
<uses-permission android:name="android.permission.VIBRATE"></uses-permission>
</manifest>
```

Start an activity when clearing Notifications

```
private void sendNotification() {
    Notification note = new Notification(R.drawable.icon, "Wake Up!",
            System.currentTimeMillis());
    PendingIntent intentToStart = PendingIntent.getActivity(this, 0,
            new Intent(this, ServiceExample.class),
            Intent.FLAG_ACTIVITY_NEW_TASK);

    note.setLatestEventInfo(this, "Sleep Time", "Time to get out of bed",
            intentToStart);
    note.number = ++count;
    note.deleteIntent = intentToStart;
    note.defaults = Notification.DEFAULT_VIBRATE;
    NotificationManager manager = (NotificationManager)
                        getSystemService(NOTIFICATION_SERVICE);
    manager.notify(NOTIFY_ID, note);
}
```

Service that Sends Notifications





Removed SD card
SD card removed. Insert a new one.

Nap Time
Go to Bed 11:02 PM

Start Service when Click button

```
public class ServiceExample extends Activity implements View.OnClickListener {

    public void onClick(View v) {
        Intent serviceIntent = new Intent(this, AvitarService.class);
        startService(serviceIntent);
    }

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button ok = (Button) findViewById(R.id.start);
        ok.setOnClickListener(this);
    }
}
```

AvitarService

```
public class AvitarService extends Service {
    private static final int NOTIFY_ID = 1123;

    public IBinder onBind(Intent arg0) {
        return null;
    }

    public void onStart(Intent intent, int startId) {
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                sendNotification();
            }
        };
        new Timer().schedule(task, 1000);
    }
```

The Notification

```
private void sendNotification() {
    Notification note = new Notification(R.drawable.icon, "Nap time!",
            System.currentTimeMillis());
    PendingIntent intentToStart = PendingIntent.getActivity(this, 0,
            new Intent(this, ServiceExample.class),
            Intent.FLAG_ACTIVITY_NEW_TASK);

    note.setLatestEventInfo(this, "Nap Time", "Go to Bed", intentToStart);
    NotificationManager manager = (NotificationManager)
                          getSystemService(NOTIFICATION_SERVICE);
    manager.notify(NOTIFY_ID, note);
}
```

Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="edu.sdsu.cs.whitney"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ServiceExample"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
            <service android:name=".AvitarService" />
    </application>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

Service with Broadcast

```
public class ServiceExample extends Activity implements View.OnClickListener {
    private Intent serviceIntent;

    private final BroadcastReceiver receiver = new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            handleBroadcast();
        }
    };

    void handleBroadcast() {
        Toast.makeText(this, "Got the message", Toast.LENGTH_SHORT).show();
    }

    public void onClick(View v) {
        serviceIntent = new Intent(this, AvitarService.class);
        startService(serviceIntent);
    }
```

Service with Broadcast

```
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button ok = (Button) findViewById(R.id.start);
        ok.setOnClickListener(this);
    }

    public void onDestroy() {
        super.onDestroy();
        stopService(serviceIntent);
    }

    public void onPause() {
        super.onPause();
        unregisterReceiver(receiver);
    }
}
```

Service with Broadcast

```
public void onResume() {
        super.onResume();
        registerReceiver(receiver, new IntentFilter(
                        AvitarService.BROADCAST_ACTION));
}
```

The Service

```
public class AvitarService extends Service {
    public static final String BROADCAST_ACTION =
            "edu.sdsu.cs.whitney.sampleBroadcast";

    private void notifyActivity() {
        Intent broadcast = new Intent(BROADCAST_ACTION);
        sendBroadcast(broadcast);
    }

    public IBinder onBind(Intent arg0) {
        return null;
    }

    public void onCreate() {
        super.onCreate();
    }
```

The Service

```
    public void onDestroy() {
        super.onDestroy();
    }

    public void onStart(Intent intent, int startId) {
        TimerTask task = new TimerTask() {
            @Override
            public void run() {
                notifyActivity();
            }
        };
        new Timer().schedule(task, 1000);
    }
}
```

Service Sending Data to Activity

The Service

```
public class AvitarService extends Service {
	private class SampleTask extends AsyncTask<Void, String, Void> {
		String[] items = { "Gautama Buddha", "Kalki", "Krishna", "Kurma",
				"Matsya", "Narasimha", "Parashurama", "Rama", "Vamana",
				"Varaha" };

		protected Void doInBackground(Void... notused) {
			for (String word : items) {
				notifyActivity(word);
				SystemClock.sleep(1000);
			}
			return (null);
		}
	}
```

Service Continued

```
SampleTask listNames;

public static final String BROADCAST_ACTION =
                                        "edu.sdsu.cs.whitney.sampleBroadcast";

private void notifyActivity(String message) {
    Intent broadcast = new Intent(BROADCAST_ACTION);
    broadcast.putExtra("name", message);
    sendBroadcast(broadcast);
}

public IBinder onBind(Intent arg0) {
    return null;
}

public void onCreate() {
    super.onCreate();
}
```

Service Continued

```
    public void onDestroy() {
        super.onDestroy();
        listNames.cancel(true);
    }

    public void onStart(Intent intent, int startId) {
        listNames = new SampleTask();
        listNames.execute();
    }
}
```

ServiceExample

```
public class ServiceExample extends Activity implements View.OnClickListener {
    private Intent serviceIntent;

    private final BroadcastReceiver receiver = new BroadcastReceiver() {
        public void onReceive(Context context, Intent intent) {
            handleBroadcast(intent.getCharSequenceExtra("name"));
        }
    };

    void handleBroadcast(CharSequence name) {
        Toast.makeText(this, name, Toast.LENGTH_SHORT).show();
    }

    public void onClick(View v) {
        serviceIntent = new Intent(this, AvitarService.class);
        startService(serviceIntent);
    }
```

Singleton for Activity Access

```
public class AvitarService extends Service {
    public static AvitarService singleton = null;

    public void onCreate() {
        super.onCreate();
        singleton = this;
    }

    public void onDestroy() {
        super.onDestroy();
        listNames.cancel(true);
        singleton = null;
    }
```

Activity

Can access service directly

```
void handleBroadcast() {
	String name = AvitarService.singleton.avitar();
	Toast.makeText(this, name, Toast.LENGTH_SHORT).show();
}



public void onDestroy() {
	super.onDestroy();
	stopService(serviceIntent);
}
```