19

Android
Intent Filters

Victor Matos
Cleveland State University

Notes are based on:
Android Developers
http://developer.android.com/index.html

Intent Filters

An Analogy: Requesting Actions Using HTTP and Android

1. The **HPPT**[1] protocol uses a number of **<Action, resource>** pairs to accomplish its work.

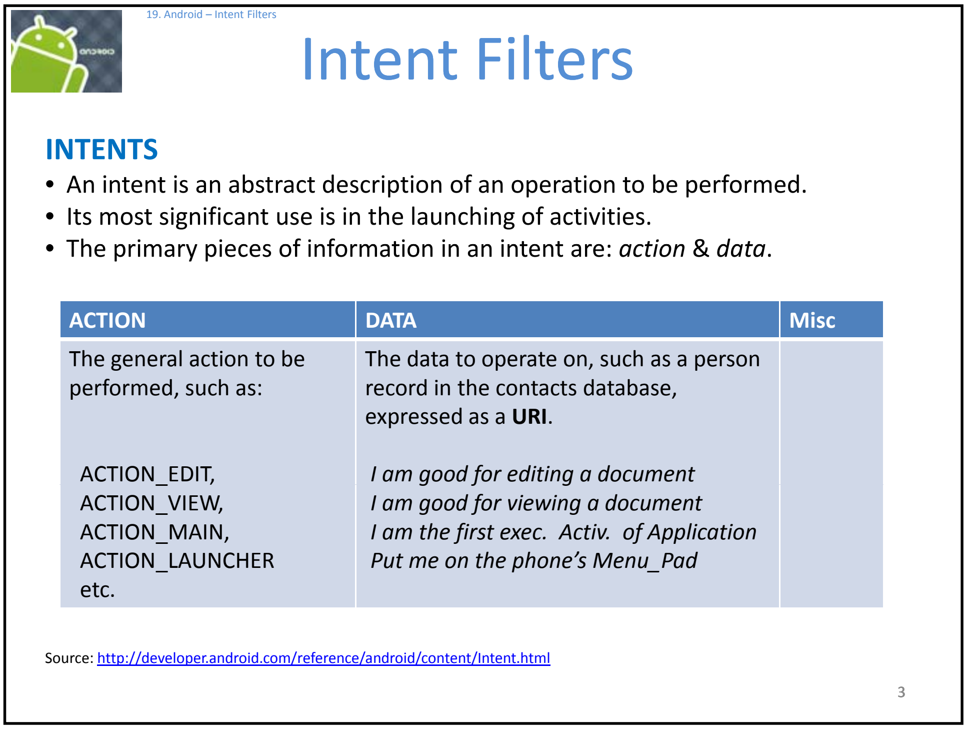
2. Some of the **HTTP** actions are the well known (and lesser known) operations: **POST, GET, PUT, DELETE, CONNECT, HEAD, OPTIONS**.

3. Android uses a mechanism quite similar to HTTP for the invocation of work to be done.

4. **INTENT** is the Android's name for the abstraction requesting actions.

5. Unlike HTTP a given Android's INTENT could be resolved in more than one potential way (for instance, we may have several SMS apps wanting to process an incoming text-message).

[1] **Source**: Hypertext Transfer Protocol -- HTTP/1.1 (1999). http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html



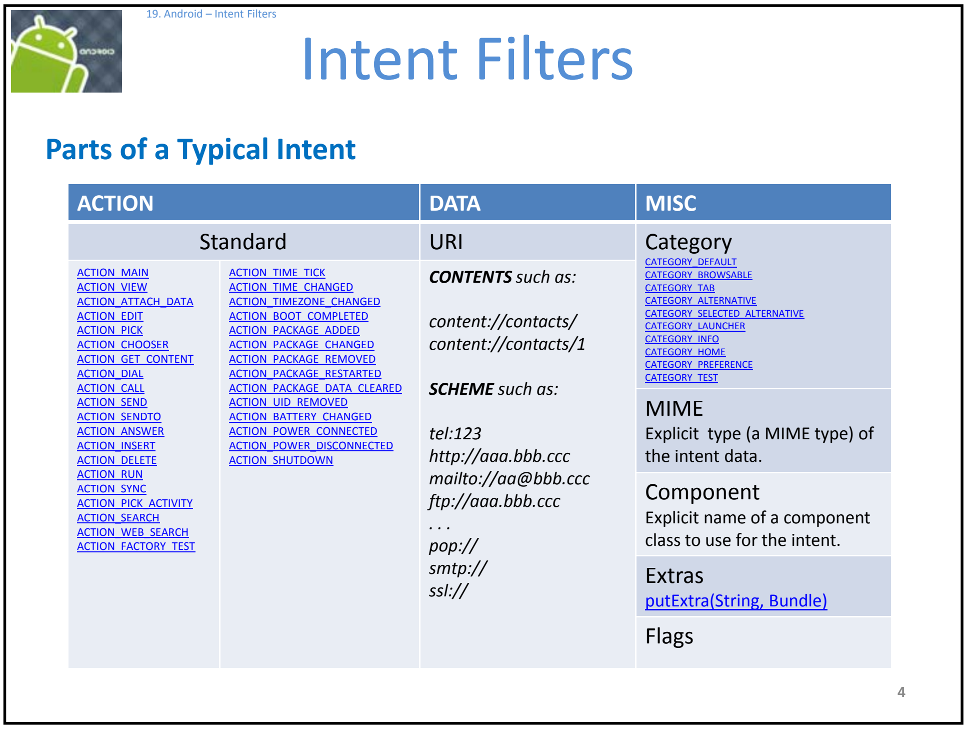
Intent Filters

INTENTS

- An intent is an abstract description of an operation to be performed.
- Its most significant use is in the launching of activities.
- The primary pieces of information in an intent are: *action* & *data*.

ACTION	DATA	Misc
The general action to be performed, such as: ACTION_EDIT, ACTION_VIEW, ACTION_MAIN, ACTION_LAUNCHER etc.	The data to operate on, such as a person record in the contacts database, expressed as a **URI**. *I am good for editing a document* *I am good for viewing a document* *I am the first exec. Activ. of Application* *Put me on the phone's Menu_Pad*	

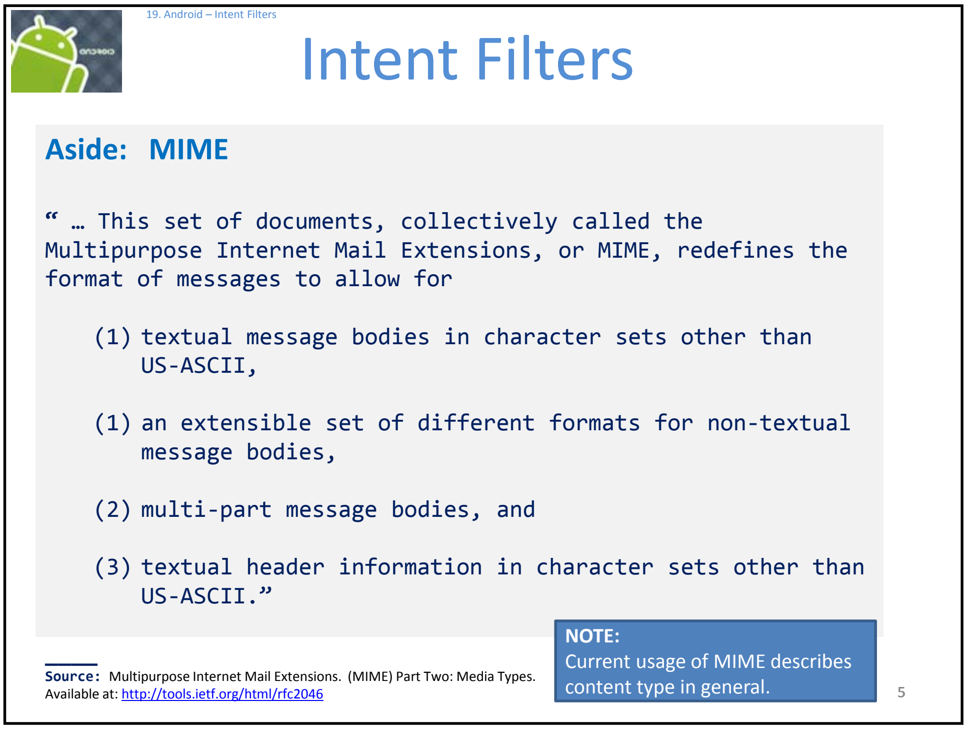
Source: http://developer.android.com/reference/android/content/Intent.html



Intent Filters

Parts of a Typical Intent

ACTION		DATA	MISC
Standard		URI	Category CATEGORY_DEFAULT CATEGORY_BROWSABLE CATEGORY_TAB CATEGORY_ALTERNATIVE CATEGORY_SELECTED_ALTERNATIVE CATEGORY_LAUNCHER CATEGORY_INFO CATEGORY_HOME CATEGORY_PREFERENCE CATEGORY_TEST
ACTION_MAIN ACTION_VIEW ACTION_ATTACH_DATA ACTION_EDIT ACTION_PICK ACTION_CHOOSER ACTION_GET_CONTENT ACTION_DIAL ACTION_CALL ACTION_SEND ACTION_SENDTO ACTION_ANSWER ACTION_INSERT ACTION_DELETE ACTION_RUN ACTION_SYNC ACTION_PICK_ACTIVITY ACTION_SEARCH ACTION_WEB_SEARCH ACTION_FACTORY_TEST	ACTION_TIME_TICK ACTION_TIME_CHANGED ACTION_TIMEZONE_CHANGED ACTION_BOOT_COMPLETED ACTION_PACKAGE_ADDED ACTION_PACKAGE_CHANGED ACTION_PACKAGE_REMOVED ACTION_PACKAGE_RESTARTED ACTION_PACKAGE_DATA_CLEARED ACTION_UID_REMOVED ACTION_BATTERY_CHANGED ACTION_POWER_CONNECTED ACTION_POWER_DISCONNECTED ACTION_SHUTDOWN	***CONTENTS** such as:* *content://contacts/* *content://contacts/1* ***SCHEME** such as:* *tel:123* *http://aaa.bbb.ccc* *mailto://aa@bbb.ccc* *ftp://aaa.bbb.ccc* *. . .* *pop://* *smtp://* *ssl://*	MIME Explicit type (a MIME type) of the intent data. Component Explicit name of a component class to use for the intent. Extras putExtra(String, Bundle) Flags



Intent Filters

Aside: MIME

“ … This set of documents, collectively called the Multipurpose Internet Mail Extensions, or MIME, redefines the format of messages to allow for

(1) textual message bodies in character sets other than US-ASCII,

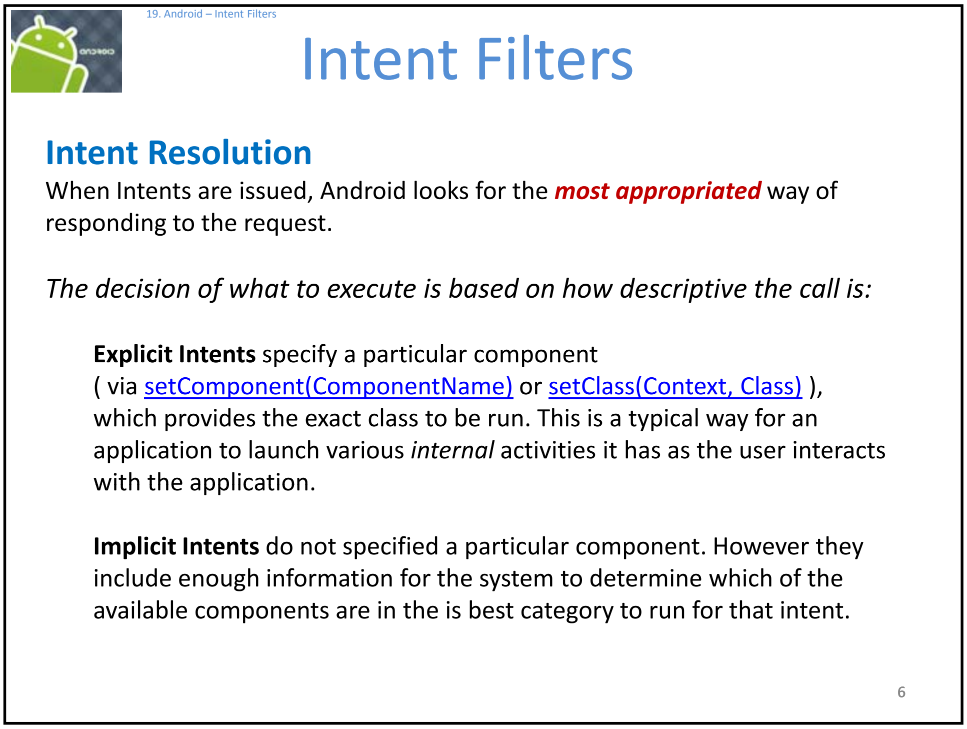
(1) an extensible set of different formats for non-textual message bodies,

(2) multi-part message bodies, and

(3) textual header information in character sets other than US-ASCII.”

Source: Multipurpose Internet Mail Extensions. (MIME) Part Two: Media Types. Available at: http://tools.ietf.org/html/rfc2046

NOTE:
Current usage of MIME describes content type in general.



Intent Filters

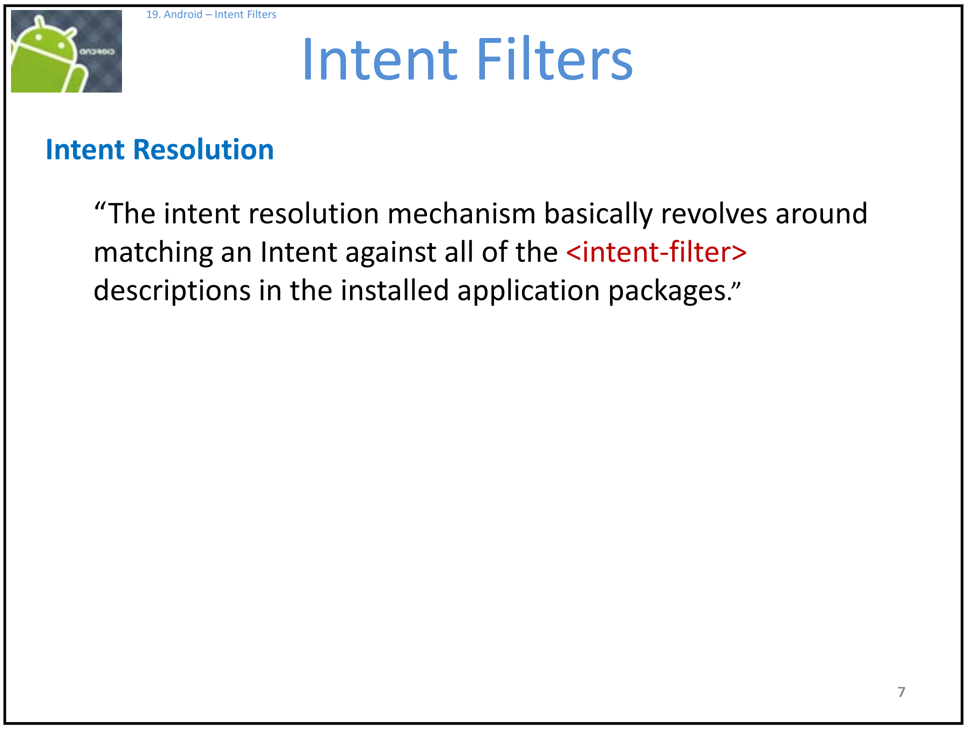
Intent Resolution

When Intents are issued, Android looks for the ***most appropriated*** way of responding to the request.

The decision of what to execute is based on how descriptive the call is:

Explicit Intents specify a particular component (via setComponent(ComponentName) or setClass(Context, Class)), which provides the exact class to be run. This is a typical way for an application to launch various *internal* activities it has as the user interacts with the application.

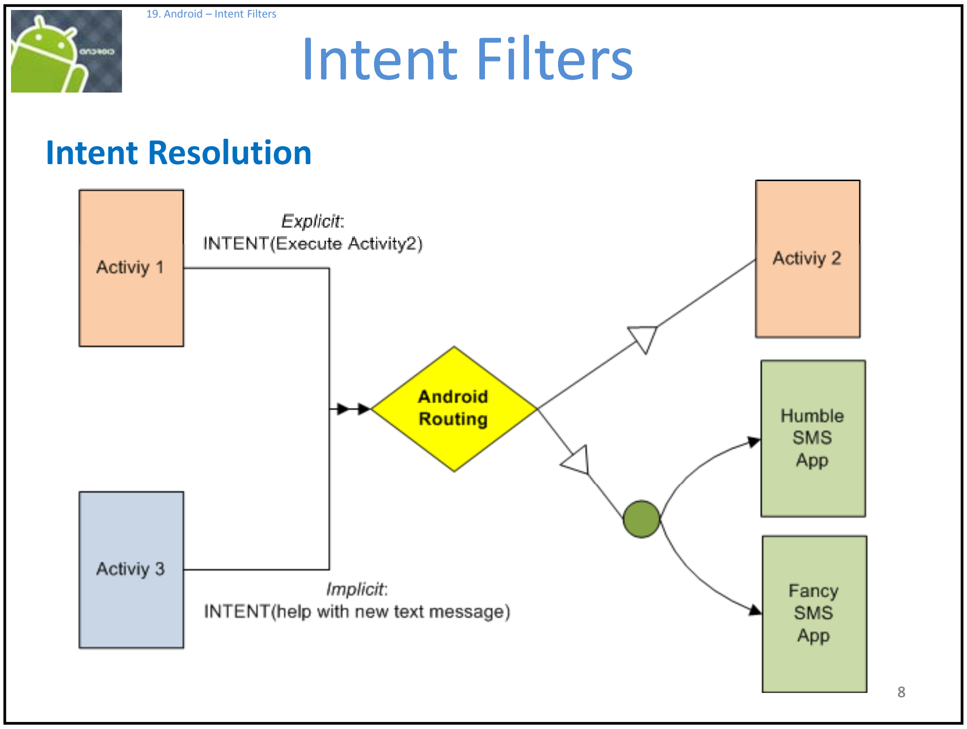
Implicit Intents do not specified a particular component. However they include enough information for the system to determine which of the available components are in the is best category to run for that intent.



Intent Filters

Intent Resolution

“The intent resolution mechanism basically revolves around matching an Intent against all of the <intent-filter> descriptions in the installed application packages.”



Intent Filters

Intent Resolution

Activiy 1

Explicit:
INTENT(Execute Activity2)

Android Routing

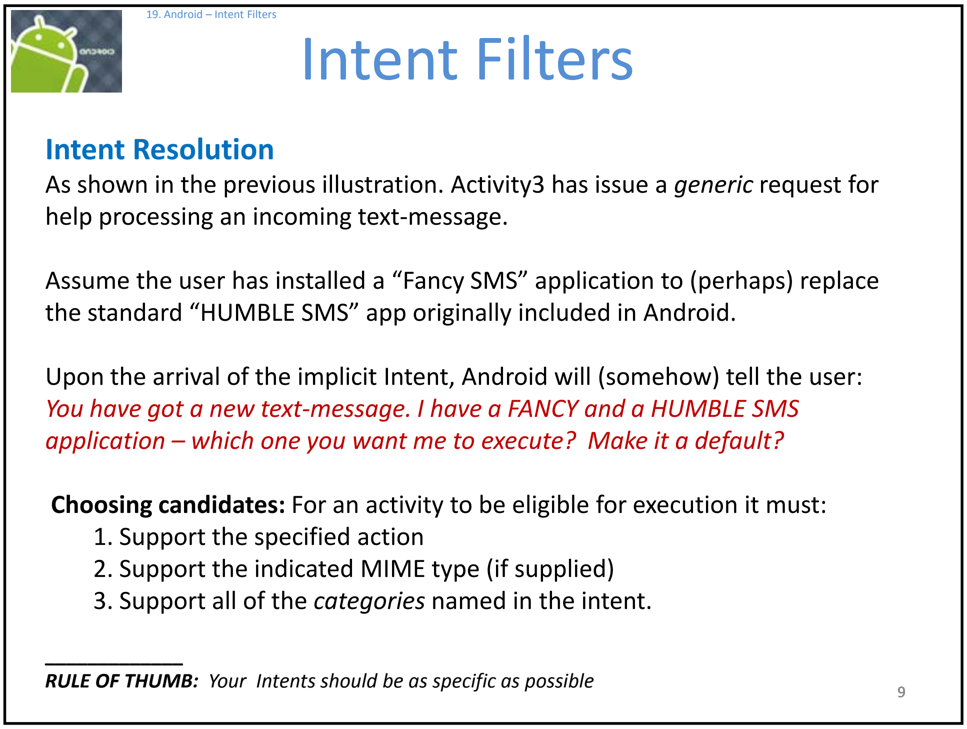
Activiy 2

Humble SMS App

Fancy SMS App

Activiy 3

Implicit:
INTENT(help with new text message)



Intent Filters

Intent Resolution

As shown in the previous illustration. Activity3 has issue a *generic* request for help processing an incoming text-message.

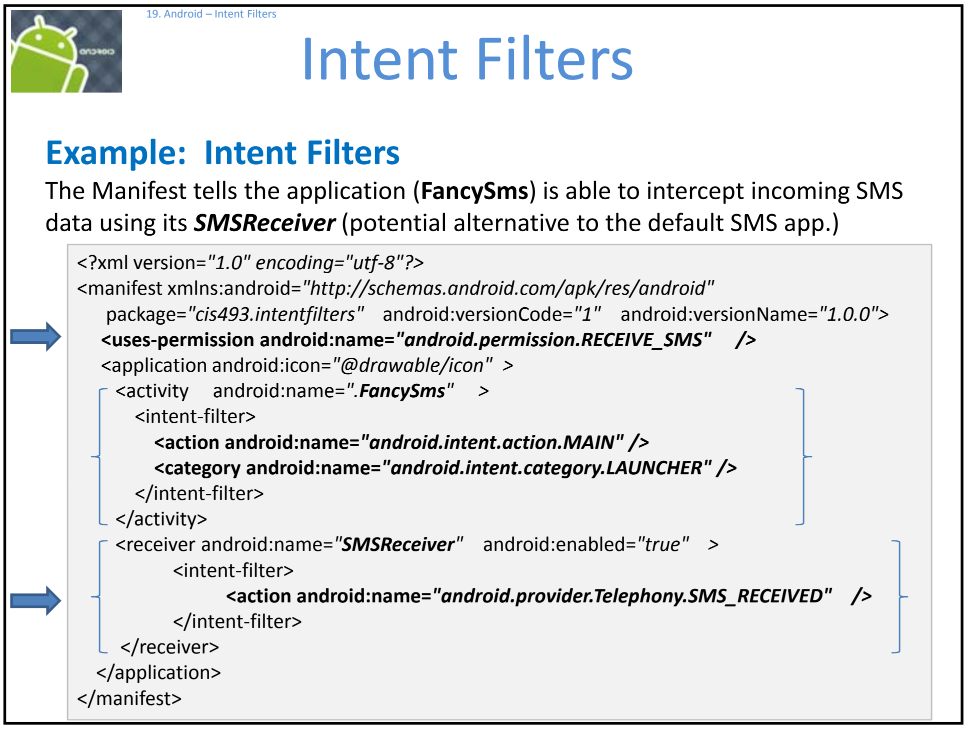
Assume the user has installed a "Fancy SMS" application to (perhaps) replace the standard "HUMBLE SMS" app originally included in Android.

Upon the arrival of the implicit Intent, Android will (somehow) tell the user:
You have got a new text-message. I have a FANCY and a HUMBLE SMS application – which one you want me to execute? Make it a default?

Choosing candidates: For an activity to be eligible for execution it must:

1. Support the specified action
2. Support the indicated MIME type (if supplied)
3. Support all of the *categories* named in the intent.

RULE OF THUMB: *Your Intents should be as specific as possible*



Intent Filters

Example: Intent Filters

The Manifest tells the application (**FancySms**) is able to intercept incoming SMS data using its ***SMSReceiver*** (potential alternative to the default SMS app.)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cis493.intentfilters"   android:versionCode="1"   android:versionName="1.0.0">
   <uses-permission android:name="android.permission.RECEIVE_SMS"    />
   <application android:icon="@drawable/icon"  >
     <activity    android:name=".FancySms"    >
        <intent-filter>
           <action android:name="android.intent.action.MAIN" />
           <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
     </activity>
     <receiver android:name="SMSReceiver"    android:enabled="true"    >
              <intent-filter>
                     <action android:name="android.provider.Telephony.SMS_RECEIVED"    />
              </intent-filter>
      </receiver>
   </application>
</manifest>
```

Intent Filters

Comments on the example:

- The application consists of two components:

1. a common Activity called **FancySms** (acting as the main routine) and
2. a background Service (BroadcastReceiver) called **SMSService**.

- The clause below indicates the application is allowed to receive SMS

  <uses-permission android:name=*"android.permission.RECEIVE_SMS"* />

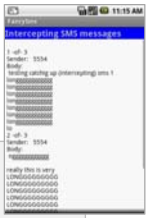
- The component **SMSService** has the filter

```
<intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"  />
</intent-filter>
```

that triggers its execution whenever a new SMS is received

- Other applications with the same filter can be also called by Android when new SMS arrives (until a DEFAULT is chosen)

Intent Filters



11:15 AM

FancySms

Intercepting SMS messages

1 -of- 3
Sender: 5554
Body:
testing catching up (intercepting) sms !

2 -of- 3
Sender: 5554
Body:

really this is very
LONGGGGGGGGGG
LONGGGGGGGGGG
LONGGGGGGGGGG
LONGGGGGGGGGG
LONGGGGGGGGGG

Example: Intercepting Incoming SMS

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
     android:id="@+id/mainLayout"
     android:layout_width="fill_parent“ android:layout_height="fill_parent"
     android:orientation="vertical"
     xmlns:android="http://schemas.android.com/apk/res/android"
>
  <TextView
     android:layout_width="fill_parent"   android:layout_height="wrap_content"
     android:textSize="20px"  android:textStyle="bold“ ndroid:background="#ff0000ff"
     android:text="Intercepting SMS messages"
     />
  <ScrollView
     android:id="@+id/myScroller1"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
     <TextView
          android:id="@+id/theMessage"
          android:layout_width="fill_parent"  android:layout_height="fill_parent"
          android:background="#ffffffff"      android:padding="4px"
          android:textSize="14px"             android:textColor="#ff000000"
          />
  </ScrollView>
</LinearLayout>
```

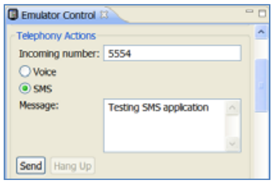
Intent Filters

Example: Intercepting Incoming SMS

Note: DDMS

Test the following application from the Eclipse's **DDMS** perspective. Select "Emulator Control" > "Telephony Actions". Set phone no. to 5554, type a message, click on Send.

Alternatively you may start another emulator and send SMS to 5554



Emulator Control

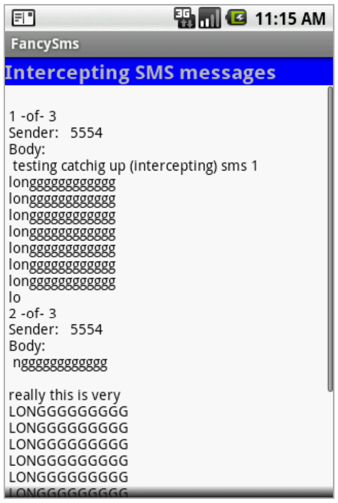
Telephony Actions

Incoming number: 5554

○ Voice

● SMS

Message: Testing SMS application



Intercepting SMS messages

```
1 -of- 3
Sender:   5554
Body:
 testing catchig up (intercepting) sms 1
longggggggggggg
longggggggggggg
longggggggggggg
longggggggggggg
longggggggggggg
longggggggggggg
longggggggggggg
lo
2 -of- 3
Sender:   5554
Body:
 nggggggggggggg

really this is very
LONGGGGGGGGG
LONGGGGGGGGG
LONGGGGGGGGG
LONGGGGGGGGG
LONGGGGGGGGG
LONGGGGGGGGG
```

Intent Filters

Example: Intercepting Incoming SMS

```java
// FancySms: main screen - displays intercepted SMS
package cis493.intentfilters;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class FancySms extends Activity {
    static TextView txtMsg;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        txtMsg = (TextView)findViewById(R.id.theMessage);
    }
}// class FancySms
```

Intent Filters

Example: Intercepting Incoming SMS

```
// SMSReceiver:  listens to broadcasted SMS_RECEIVED signals
package cis493.intentfilters;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.gsm.SmsMessage;
import android.widget.Toast;

public class SMSReceiver extends BroadcastReceiver {

@Override
public void onReceive(Context context, Intent intent) {
// Android saves in a bundle the current text-message
// under name "pdus" and type: Object[]. Later we cast to
// SmsMessage[]. Jargon pdu stands for "protocol data unit"
Bundle bundle = intent.getExtras();
```

Intent Filters

Example: Intercepting Incoming SMS

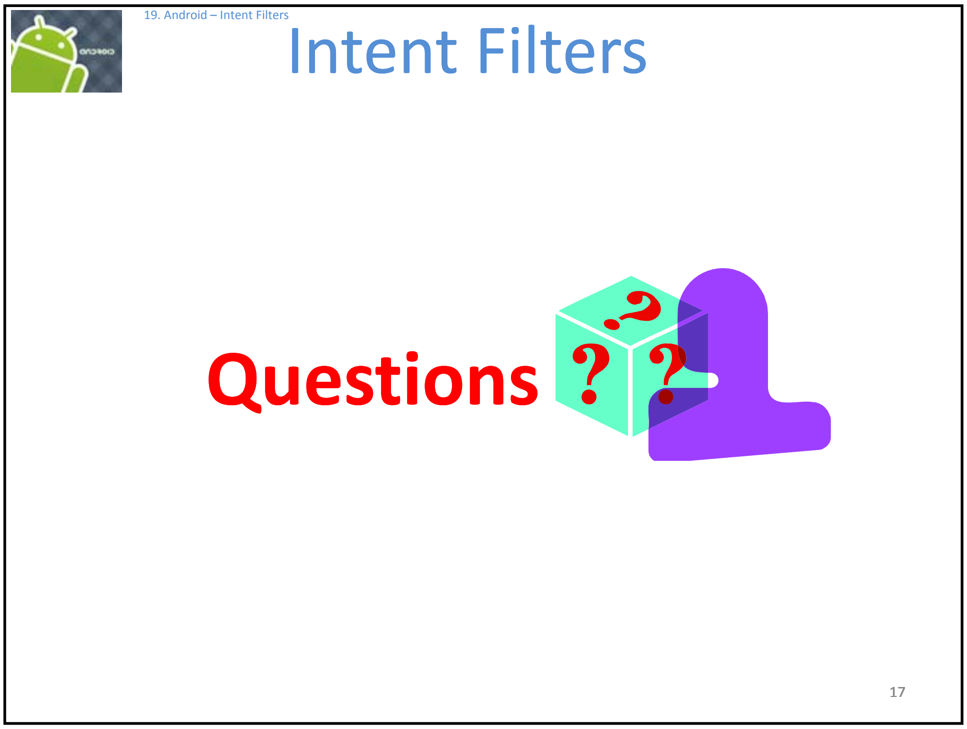
```
    Object messages[] = (Object[]) bundle.get("pdus");
    SmsMessage smsMessage[] = new SmsMessage[messages.length];

    // Note: long sms are broken and transmitted into various pieces
    String msg = "";
    int smsPieces = messages.length;

    for (int n = 0; n < smsPieces; n++) {
     smsMessage[n] = SmsMessage.createFromPdu((byte[]) messages[n]);
     // grab all pieces of the intercepted sms
     msg  += "\n" + (n + 1) + " -of- " + smsPieces + "\n"
          + "Sender:\t"  + smsMessage[n].getOriginatingAddress() + "\n"
          + "Body:  \n " + smsMessage[n].getMessageBody();
    }

    // show first part of intercepted (current) message
    Toast toast = Toast.makeText(context, "FANCY >>> Received SMS: "
                 + smsMessage[0].getMessageBody(), Toast.LENGTH_LONG);
    toast.show();

    cis493.intentfilters.FancySms.txtMsg.setText(msg);
    }
}// class SMSReceiver
```

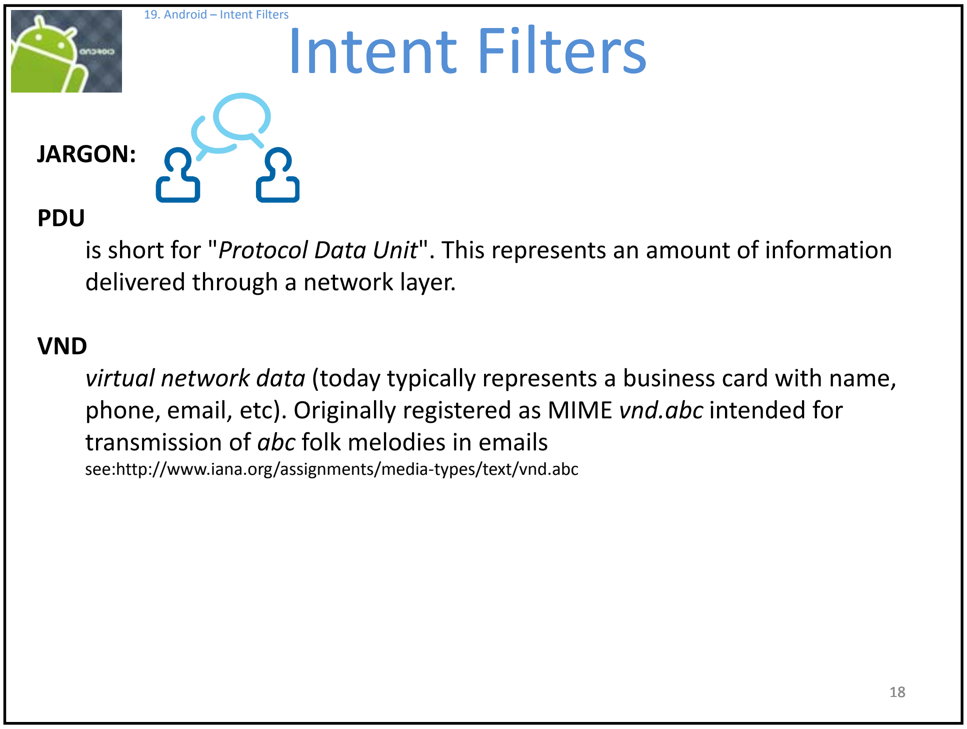


19. Android – Intent Filters

Intent Filters

Questions

17



19. Android – Intent Filters

Intent Filters

JARGON:

PDU

is short for "*Protocol Data Unit*". This represents an amount of information delivered through a network layer.

VND

virtual network data (today typically represents a business card with name, phone, email, etc). Originally registered as MIME *vnd.abc* intended for transmission of *abc* folk melodies in emails

see:http://www.iana.org/assignments/media-types/text/vnd.abc

18