

# ANDROID platforma uvod u razvoj aplikacija

Mobilni i distribuirani informacijski sistemi

*Mr Bratislav Predić*

2012. godina



# Šta je Android?

- Android je open-source softverska platforma razvijena od strane *Google-a* i *Open Handset Alliance*
- Danas se uglavnom koristi kao OS mobilnih telefona
- Ima perspektivu da se koristi kao generalni OS kućne elektronike
- Android se sastoji od kompletnog seta softverskih komponenti
  - Operativnog sistema
  - Middleware-a
  - Ugrađenih korisničkih mobilnih aplikacija i
  - Software market-a



# Šta je Android?

- Android je softversko okruženje
- NIJE hardverska platforma
- Android uključuje
  - Kernel OS zasnovan na Linux-u
  - Bogat i atraktivan korisnički interfejs
  - Podršku za funkcionalnost mobilnog telefona
  - Korisničke aplikacije
  - Biblioteke
  - Framework za razvoj aplikacija
  - Multimedijalna podrška itd.
- Aplikacije za Android se razvijaju u Java programskom jeziku



# Android i konkurencija



1. Apple Inc.
2. Microsoft
3. Nokia
4. Palm
5. Research In Motion
6. Symbian

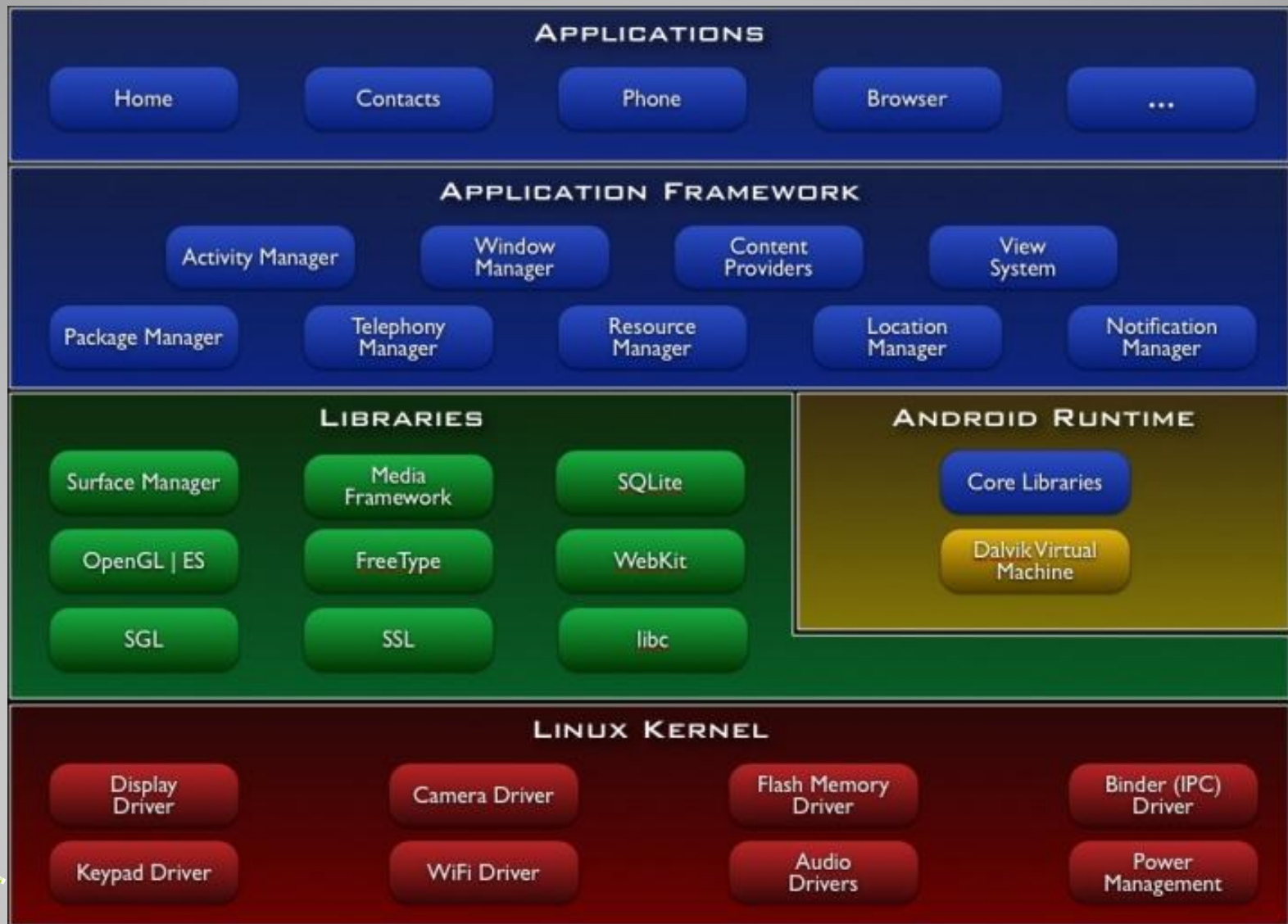


# Android komponente

- Application framework omogućava code reuse
- Dalvik Java virtuelna mašina prilagođena mobilnim uređajima
- Integriran Web browser zasnovan na WebKit engine-u
- Optimizovana grafika – 2D i 3D zasnovana na OpenGL ES specifikaciji sa hardverskom akceleracijom
- Podrška za medija formate (MPEG4, H.264 ...)
- GSM telefonija
- Kamera, GPS, kompas, akceleracioni senzor
- Bogato razvojno okruženje u Eclipse-u



# Android komponente



# Zašto Linux?

- Linux kernel je dokazana platforma
- Pouzdanost je ključna karakteristika u mobilnoj telefoniji
- Linux omogućava sloj apstrakcije hardvera
  - Viši slojevi ostaju nepromenjeni kada se menja hardverska platforma
- Kada se pojavi novi hardver na tržištu dražveri se razvijaju kao za bilo koju drugu Linux platformu
- Korisničke aplikacije kao i Android “sistemske” aplikacije se pišu u Javi i prevode u byte code



# Čemu još jedna Java VM

- Android byte-code interpretira tokom izvršenja aplikacije u Dalvik virtualnoj mašini
- Android byte-code je logički ekvivalentan Java byte-code-om
- Custom virtualna mašina ne potpada pod licencna ogranišenja kompanije Sun
- Otvorena platforma koju je moguće nezavisno unapređivati i menjati po potrebi





# Elementni Android aplikacija

- **Intent** – opisuje *šta želimo da uradimo*
- Primeri
  - Želim da prikažem kontakte u imeniku
  - Želim da prikazem web site
  - Želim da obavim glasovni poziv itd.
- Intent-i omogućavaju navigaciju između elemenata Android aplikacija i ključni su aspekt Android programiranja
- Intent se sastoji od
  - Željene akcije ili servisa
  - Podataka
  - Kategorije komponente koja će obraditi intent



# Intent i IntentFilter

- **IntentFilter** je trigger kojim aplikacija registruje sposobnost da obradi neki intent
- Intent-i se najčešće koriste kako bi startovali aktivnost (Activity) i praktično služe kao veza za navigaciju između aktivnosti
- Intent može biti
  - *Eksplicitan* – imenuje komponentu (konkretnu aktivnost) koja treba da ga obradi
  - *Implicitan* – ne imenuje komponentu i obično se koristi za aktiviranje komponenti drugih aplikacija
- Intent objekat se prosleđuje
  - `Context.startActivity()`, `Context.startService()`, `Context.bindService()`



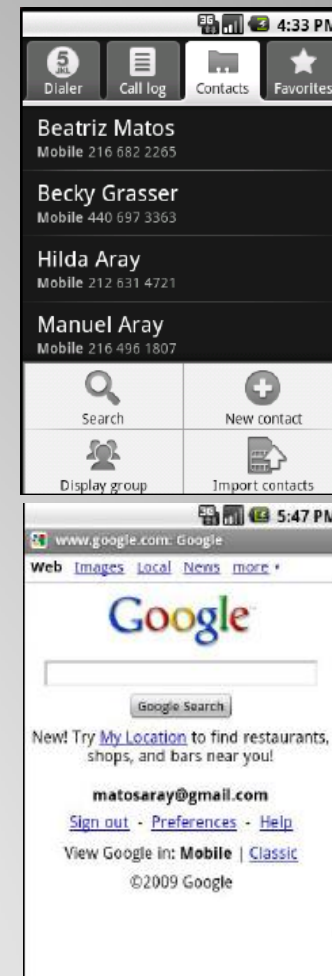
# Intent primeri

- Kreiramo Intent koji aktivira ugrađeni task za prikaz kontakata na telefonu

```
Intent myIntent = new Intent(  
    Intent.ACTION_VIEW,  
    Uri.parse("content://contacts/people"));  
  
startActivity(myIntent);
```

- Prikaz Web stranice

```
Intent myIntent = new Intent(  
    Intent.ACTION_VIEW,  
    Uri.parse("http://www.google.com"));  
  
startActivity(myIntent);
```



# Intent primeri

- Iniciramo glasovni poziv

```
Intent myIntent = new Intent(  
    Intent.ACTION_VIEW,  
    Uri.parse("tel:/216 555-1234"));  
  
startActivity(myIntent);
```



- Kompletan kod

```
package matos.cis493;  
import android.app.Activity;  
import android.content.Intent;  
import android.net.Uri;  
import android.os.Bundle;  
  
public class AndDemo1 extends Activity {  
    /** show contact list */  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        Intent myIntent = new Intent( Intent.ACTION_VIEW, Uri.parse( "content://contacts/people"));  
        startActivity(myIntent);  
    }  
}
```



# IntentFilter

- IntentFilter povezuje Intent i aplikaciju
- IntentFilter se može vezati za akciju, podatke ili oba
- IntentFilter sadrži i kategoriju
  - Kategorija CATEGORY\_LAUNCHER znači da će ta aktivnost biti vidljiva na Home ekranu (najčešće je to glavna aktivnost aplikacije koju korisnik startuje)
- Kada se Intent pošalje sistem ispituje aktivnosti, servise i BroadcastReceiver-e i šalje Intent najadekvatnijoj komponenti
- Ove komponente mogu imati jedan ili više IntentFilter-a



# IntentFilter

- Eksplicitni Intent se UVEK isporučuje imenovanoj aktivnosti (komponenti)
- Implicitni Intent se isporučuje samo komponentama čije filtre zadovoljava
- Primer filtra definisanog u manifest-u aplikacije

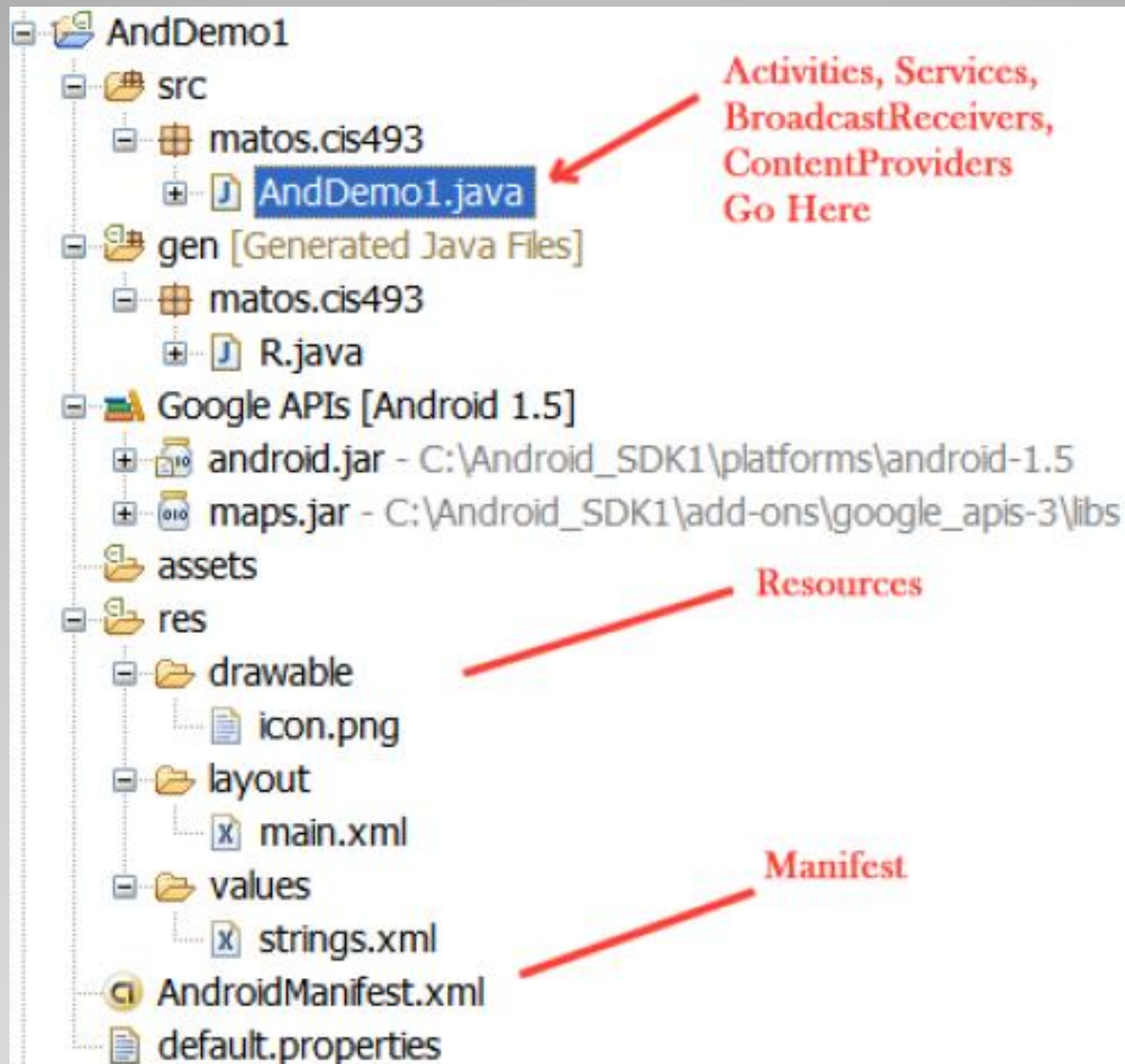
```
<intent-filter ... >
  <action android:name="code android.intent.action.MAIN" />
  <category android:name="code android.intent.category.LAUNCHER" />
  <category android:name="android.intent.category.BROWSABLE" />
  <data android:type="video/mpeg" android:scheme="http" ... />
  <data android:type="audio/mpeg" android:scheme="http" ... />
  ...
</intent-filter>
```

# Android aplikacije

- Svaka Android aplikacija se izvršava u svom procesu
- Aplikacija se može sastojati od različitog broja
  - Aktivnosti (Activity)
  - Servisa (Service)
  - Broadcast prijemnika (BroadcastReceiver)
  - Izvora sadržaja (ContentProvider)



# Struktura Android projekta





# Android Service

- Service je komponenta koja se izvršava bez interakcije sa korisnikom (nema UI – izvršava se u pozadini)
- Svaki servis se izvršava u glavnom thread-u svog procesa
- Ako servis treba da obavi neku blokirajuću operaciju (komunikacija preko mreže) treba da startuje novi thread
- Svaki servis treba da ima `<service>` tag u manifestu aplikacije
- Servisima se upravlja sa
  - `Context.startService()` i `Context.stopService()`



# Primer Android servisa

```
package matos.service;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;

public class Service1 extends Service implements Runnable {
    private int counter = 0;
    @Override
    public void onCreate() {
        super.onCreate();
        Thread aThread = new Thread(this);
        aThread.start();
    }

    public void run() {
        while (true) {
            try {
                Log.i("service1", "service1 firing : # " + counter++);
                Thread.sleep(10000); //this is where the heavy-duty computing occurs
            } catch (Exception ee) {
                Log.e("service1", ee.getMessage());
            }
        }
    }
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```



# Startovanje servisa

```
package matos.service;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
public class Service1Driver extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // invoking the service
        Intent service1Intent = new Intent( this, Service1.class );
        startService( service1Intent );
        // do some work here....
    }
}
} // Service1Driver
```



# Manifest aplikacije sa servisom

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="matos.service"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Service1Driver"
            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="Service1" android:enabled="true" >
        </service>
    </application>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

# BroadcastReceiver

- Aplikacija koja želi da dobije obaveštenje o globalnim događajima (dolazni poziv, SMS) mora da se registruje kao BroadcastReceiver
- Ako je aplikacija registrovana u manifest-u automatski se startuje po detekciji događaja
- Kao i servisi i BroadcastReceiver-i nemaju UI
- Metod *onReceive()* BroadcastReceiver-a mora da se izvrši brzo ili da pošalje zahtev servisu



# BroadcastReceiver primer

```
package matos.broadcastreceiver;

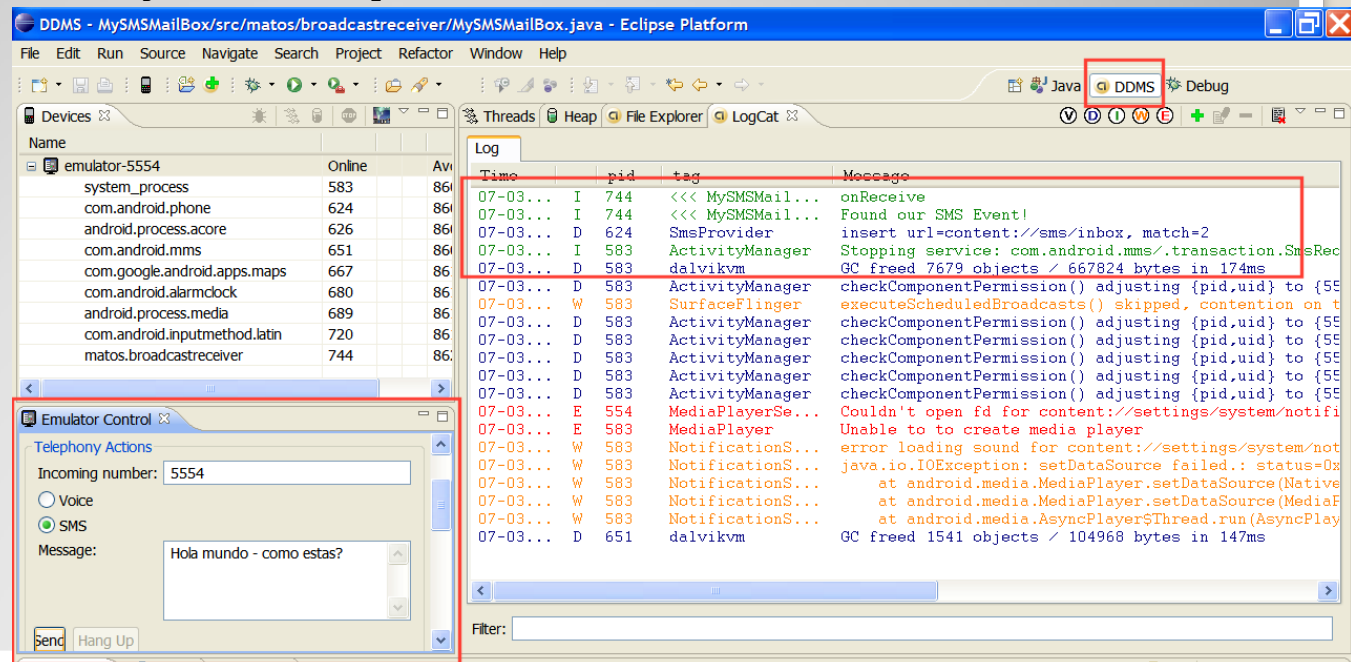
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.util.Log;
import android.app.Activity;
import android.os.Bundle;

public class MySMSMailBoxextends Activity {
    // intercepts reception of new text-messages
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // define instance of local broadcast receiver
        MySMSMailBoxReceivermySmsReceiver= new MySMSMailBoxReceiver();
        // receiver's filter will accept event: ...SMS_RECEIVED
        IntentFilterfilter = new IntentFilter("android.provider.Telephony.SMS_RECEIVED");
        // tell Android OS this receiver is ready to go
        registerReceiver(mySmsReceiver, filter);
    }
}
```



# BroadcastReceiver primer

```
// this is the custom made broadcast receiver. Its onReceive method
// is fired when the filter matches the SMS_RECEIVED event
public class MySMSMailBoxReceiver extends BroadcastReceiver {
    public static final String tag = "<<< MySMSMailBox>>>";
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i(tag, "onReceive");
        // checking global event signaling arrival of text-message
        if (intent.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {
            Log.i(tag, "Found our SMS Event!");
            // you have intercepted the SMS
            // do something interesting with it. Bye!
        }
    }
} // onReceive
} // BroadcastReceiver
```



DDMS - MySMSMailBox/src/matos/broadcastreceiver/MySMSMailBox.java - Eclipse Platform

File Edit Run Source Navigate Search Project Refactor Window Help

Log

Time	pid	tag	Message
07-03...	I 744	<<< MySMSMail...	onReceive
07-03...	I 744	<<< MySMSMail...	Found our SMS Event!
07-03...	D 624	SmsProvider	insert url=content://sms/inbox, match=2
07-03...	I 583	ActivityManager	Stopping service: com.android.mms/.transaction.SmsRec
07-03...	D 583	dalvikvm	GC freed 7679 objects / 667824 bytes in 174ms

Emulator Control

Telephony Actions

Incoming number: 5554

☐ Voice

☒ SMS

Message: Hola mundo - como estas?

Send Hang Up

# BroadcastReceiver primer

- Manifest primera

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="matos.broadcastreceiver"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MySMSMailBox" 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-permission android:name="android.permission.RECEIVE_SMS" />
    <receiver android:name="MySMSMailBoxReceiver" >
        <intent-filter>
            <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
        </intent-filter>
    </receiver>
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```



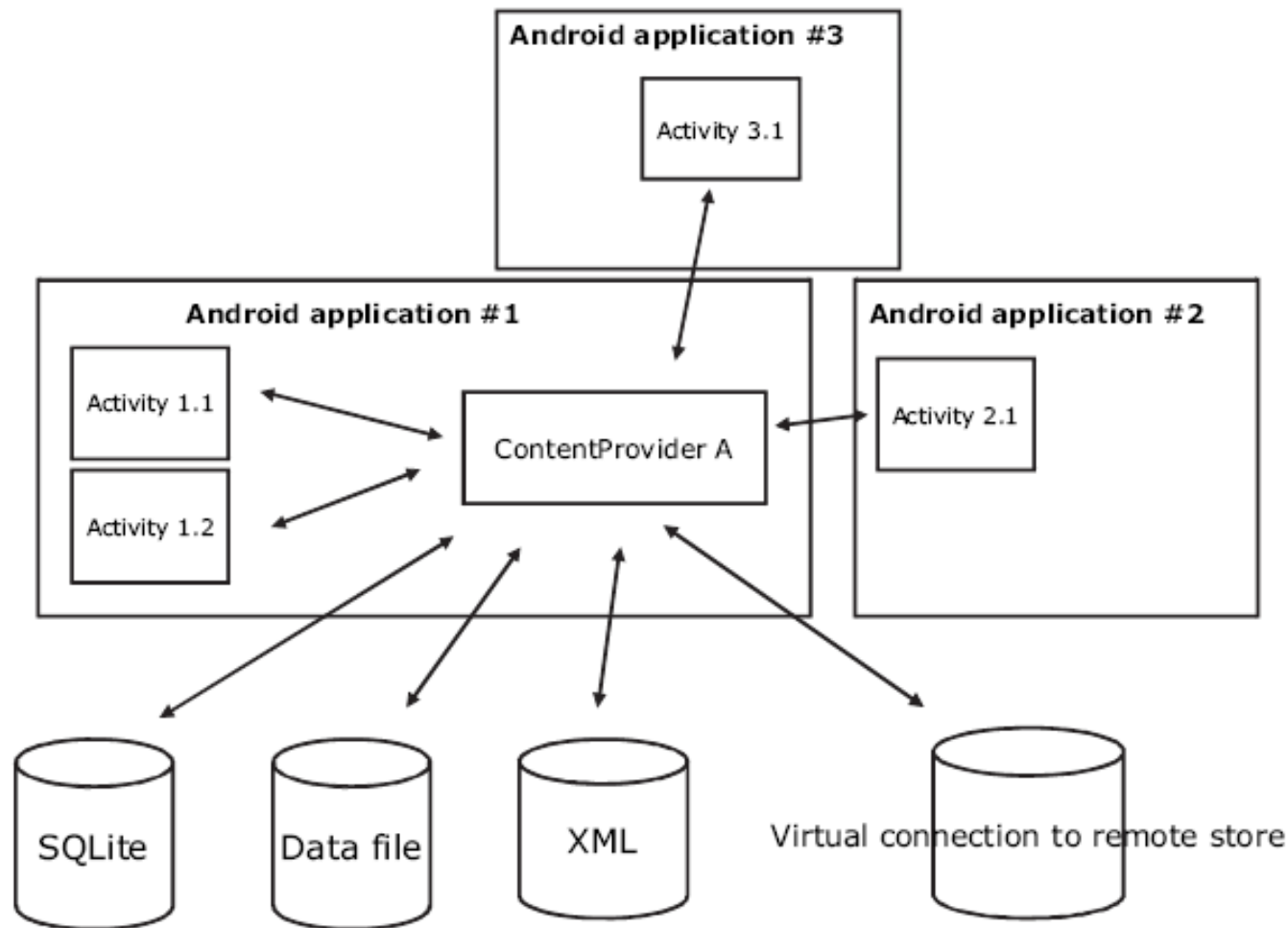


# ContentProvider

- Perzistira podatke i omogućava pristup tim podacima
- Ovo je jedini način kako različite Android aplikacije mogu razmenjivati podatke
- ContentProvider može pružiti podatke Activity ili Service softverskim komponentama
- ContentProvider može koristiti bilo koji metod smeštanja podataka
  - Fajlove, SQLite bazu ili podaci uopšte ne moraju biti perzistentni (samo u memoriji)



# ContentProvider



**Figure 1.6** The content provider is the data tier for Android applications and is the prescribed manner in which data is accessed and shared on the device.

# ContentProvider

- Model podataka ContentProvider-a je tabela
- Svaki ContentProvider definiše URI koji se koristi za pristup tom provider-u
- Ukoliko provider ima više skupova podataka postoji URI za svaki skup
- URI počinje sa "content://"
- Postoje neki predefinisani URI

`android.provider.Contacts.Phones.CONTENT_URI`

`android.provider.Contacts.Photos.CONTENT_URI`

`android.provider.CallLog.Calls.CONTENT_URI`

`android.provider.Calendar.CONTENT_URI`

- Query vraća Cursor koji se koristi za šetanje kroz record-e



# Android emulator



# Kontrola Android emulatora

Keyboard	OS function
Escape	Back button
Home	Home button
F2, PageUp	Menu (Soft-Left) button
Shift-F2, PageDown	Start (Soft-Right) button
F3	Call/Dial button
F4	Hangup / EndCall button
F5	Search button
F7	Power button
Ctrl-F3, Ctrl-KEYPAD_5	Camera button
Ctrl-F5, KEYPAD_PLUS	Volume up button
Ctrl-F6, KEYPAD_MINUS	Volume down button
KEYPAD_5	DPad center
KEYPAD_4	DPad left
KEYPAD_6	DPad right
KEYPAD_8	DPad up
KEYPAD_2	DPad down
F8	toggle cell network on/off
F9	toggle code profiling (when -trace option set)
Alt-ENTER	toggle FullScreen mode
Ctrl-T	toggle trackball mode
Ctrl-F11, KEYPAD_7	switch to previous layout
Ctrl-F12, KEYPAD_9	switch to next layout



# Android emulator

- Emulator simulira
  - ARM CPU
  - LCD displej
  - Tastaturu / DPad, Phone dugmiće
  - Zvuk sa ulazom i izlazom
  - Flash memoriju
  - GSM model sa simuliranom SIM karticom
  - GPS prijemnik



# Android emulator

- Na development računar u možemo kreirati proizvoljan broj Android Virtual Device (AVD) emulatora različitih karakteristika
- Prikaz liste kreiranih AVD

```
android list avd
```


- Startovanje AVD emulatora

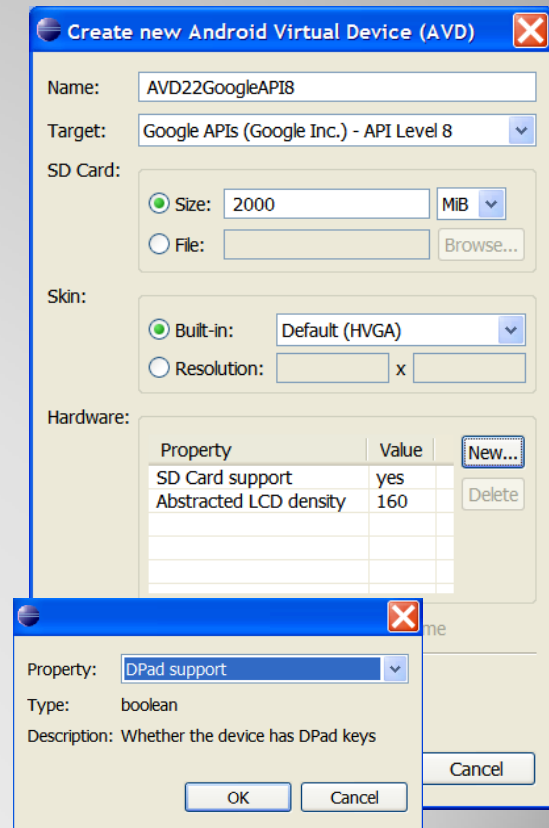
```
emulator -avd <AVDName>
```

- AVD se sastoji od
  - Hardverskog profila (kamera, QWERTY tastatura, DPAD ...)
  - Image-a sistema
  - Ostalih opcija (rezolucija ekrana, SD kartice ...)



# Android emulator

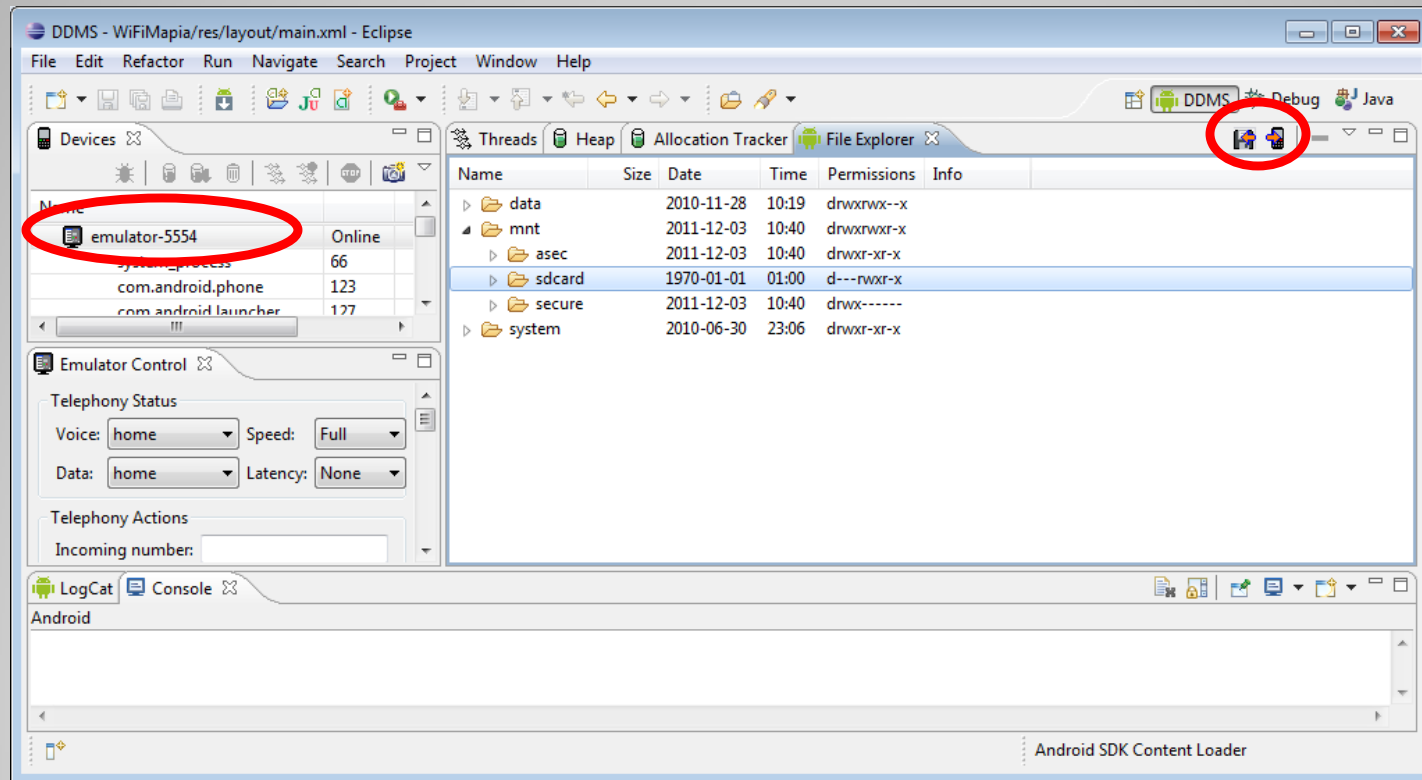
- AVD se može kreirati iz Eclipse okruženja
  - Main Menu (AVD Manager  ) > Virtual Device > New
- Osnovni parametri AVD-a
  - Naziv
  - API (verzija Android-a)
  - Veličina SD kartice
  - Rezolucija ekrana
  - Hardverske karakteristike
    - DPad podrška
    - Podrška za senzore (akceleracioni)
    - LCD density
    - ...





# Android emulator

- Rad sa simuliranom SD karticom
- Koristimo DDMS (Dalvik Debug Monitor Service)



# Android emulator – OS shell

- Možemo da se logujemo na Android Linux OS shell i u tekstualnom režimu izvršavamo komande
- Koristimo ADB (Android Debug Shell)

```
adb shell
```

- Ukoliko imamo više emulatora startovanih

```
adb devices
```

List of devices

```
attachedemulator-5554
```

```
deviceemulator-5556
```

```
deviceHT845GZ45737 device
```

```
adb -s emulator-5554 shell
```

```
C:\WINDOWS\system32\cmd.exe - adb shell
C:\>adb shell
# ls -l
ls -l
drwxrwxrwt root    root    2008-12-04 07:53 sqlite_stmt_journals
drwxrwx--- system cache    2008-12-04 07:53 cache
d---rwxrwx system system   2008-12-04 08:24 sdcard
lrwxrwxrwx root    root    2008-12-04 07:53 etc -> /system/etc
-rwxr-x--- root    root    98260 1969-12-31 19:00 init
-rwxr-x--- root    root    1564 1969-12-31 19:00 init.goldfish.rc
-rwxr-x--- root    root    8630 1969-12-31 19:00 init.rc
drwxrwx-x system system    2008-09-22 16:44 data
drwxr-xr-x root    root    2008-09-22 16:41 system
drwxr-xr-x root    root    1969-12-31 19:00 sys
dr-xr-xr-x root    root    1969-12-31 19:00 proc
drwxr-x--- root    root    1969-12-31 19:00 sbin
-rw-r--r-- root    root    93 1969-12-31 19:00 default.prop
drwx----- root    root    1969-12-31 19:00 root
drwxr-xr-x root    root    2008-12-04 07:53 dev
# df
df
/dev: 47284K total, 0K used, 47284K available (block size 4096)
/sqlite_stmt_journals: 4096K total, 0K used, 4096K available (block size 4096)
/system: 65536K total, 42320K used, 23216K available (block size 4096)
/data: 65536K total, 21692K used, 43844K available (block size 4096)
/cache: 65536K total, 1156K used, 64380K available (block size 4096)
/sdcard: 258064K total, 22941K used, 235123K available (block size 512)
# cd sdcard
cd sdcard
# ls -l
ls -l
d---rwxrwx system system    2008-12-04 08:26 Pictures
---rw-rw- system system    5239976 2008-12-03 21:24 Amarcord.mp3
---rw-rw- system system    6853190 2008-12-03 21:26 El Platanal de Bartolo.mp3
```

# Android emulator i realni uređaji

- Sa realnim uređajima se radi na skoro identičan način kao sa emulatorima
- Telefon se USB kablom povezuje sa development računarom
- Kada pristupimo OS shell-u možemo dobiti shell sa oznakom “#” (administratorski pristup)
- Kao i na standardnom Linux OS-u možemo da probamo da dobijemo root pristup su
- Na standardnim komercijalno dostupnim verzijama Android telefona korisnik nema “root” (administratorski pristup) OS-u



# Pull aplikacija

- Instalirane aplikacije možemo skinuti sa root-ovanog hardvera i instalirati na emulatoru
- Prvo povežemo realan telefon

```
adb devices
```

```
adb -s HT845GZ45737 pull data/app/theInstalled.apk  
c:/theInstalled.apk
```

- Sada povežemo emulator

```
adb -s emulator-5554 install c:\theInstalledApp.apk
```

```
adb -s emulator-5554 uninstall data/app/theInstalled.apk
```



# Komande shell-a

- Android shell podržava neke od standardnih Linux shell komandi

```
ls..... show directory (alphabetical order)
mkdir..... make a directory
rmdir..... remove directory
rm-r ..... to delete folders with files
rm..... remove files
mv..... moving and renaming files
cat ..... displaying short files
cd..... change current directory
pwd..... find out what directory you are in
df..... shows available disk space
chmod..... changes permissions on a file
date ..... display date
exit ..... terminate session
```



# Emulator – slanje SMS poruka

- Sve eksterne događaje (SMS poruke, glasovne pozive, GPS koordinate itd.) možemo simulirati iz konzole ili korišćenjem DDMS GUI alata
- Iz konzole

```
adb devices
```

```
telnet localhost 5554
```

```
sms send <broj_telefona_posiljaoca> <tekst_poruke>
```

```
C:\WINDOWS\system32\cmd.exe
```

```
Microsoft Windows XP [Version 5.1.2600]  
(C) Copyright 1985-2001 Microsoft Corp.
```

```
c:\Android\tools>telnet localhost 5554
```

```
Telnet localhost
```

```
Android Console: type 'help' for a list of commands
```

```
OK  
sms send 5551122 don't forget we have a ballroom class tonight at 8:30p  
OK
```

Beatriz Aurenty: don't forg



3G 2:45 AM  
Beatriz Aurenty <216-555-1122>



**Beatriz Aurenty:** don't forget we have a ballroom class tonight at 8:30pm

Sent: 2:44AM

Type to compose

Send

# Emulator – glasovni pozivi

- Na skoro identičan način kao i simulacija SMS poruka
- Iz konzole

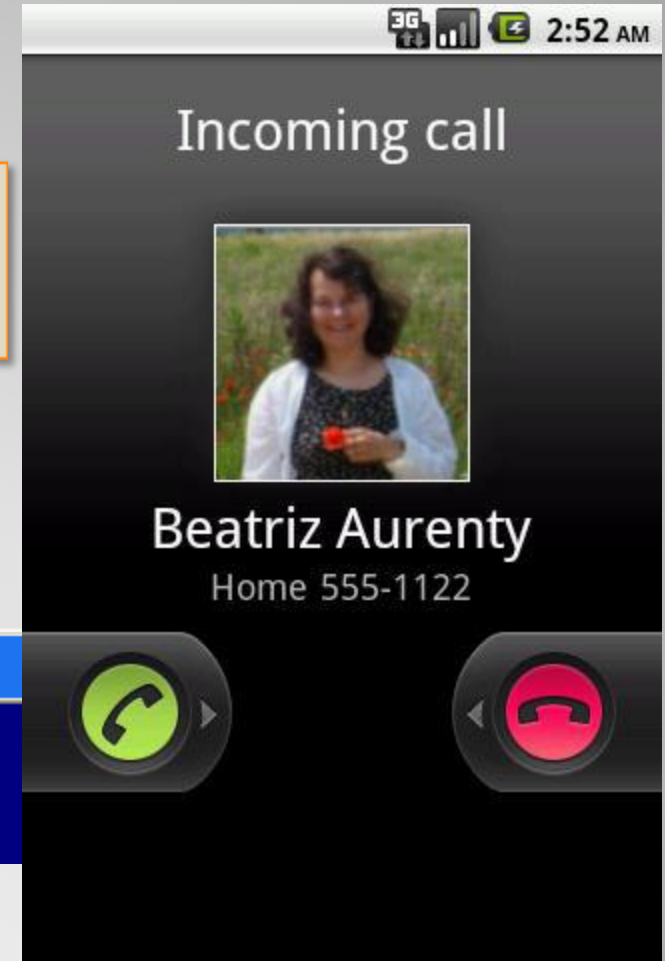
```
adb devices
telnet localhost 5554
gsm call <broj_telefona_sa_koji_zove>
```

```
C:\WINDOWS\system32\cmd.exe
```

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\Android\tools>telnet localhost 5554_
```

```
Telnet localhost
```

```
Android Console: type 'help' for a list of commands
OK
gsm call 5551122
OK
```



# Emulator – GPS pozicioniranje

- Emulator omogućava zadavanje koordinata i testiranje location-aware aplikacija
- Kontrola iz konzole

```
telnet localhost 5554
geo fix -121.45356 46.51119 4392
ili
geo nmea $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
```

- Prihvata GPRMC i GPGGA NMEA rečenice
- Kontrola iz DDMS-a
- Dve mogućnosti
  - Ručno unošenje koordinata
  - Playback GPX (GPS eXchange) ili KML (Keyhole Markup Language) snimljenih ruta

