Android

Ing. Ladislav Pešička

Historie

- 2005 Google kupuje Android, Inc.
- 2007 ohlášena Open Handset Alliance
  Android je open sourced
- 2008 Android SDK 1.0 vydáno
  telefon G1 od HTC
- 2009 nové verze 1.5, 1.6, 2.0, 2.1,
  > 20 zařízení na Androidu
- 2010 verze 2.2 (Froyo), > 60 zařízení
- 2011 Barcelona Mobile Congress 2011
  převládala zelená barva ☺
- 2011 verze 4.0 (Android Ice Cream Sandwitch)

Vývoj

- Android SDK zdarma
  - včetně emulátoru zařízení
- Eclipse + ADT plugin

- celá řada dalších vývojových nástrojů a prostředí, např. Adobe AIR, App Inventor http://appinventor.googlelabs.com/about/

Emulátor

klávesové zkratky:

C:\sw\android-sdk-windows\tools>emulator -help-keys

Klávesová zkratka	činnost
Ctrl+F11, Ctrl+F12	Emulátor na výšku / šířku
Home	Tlačítko Home
Escape	Tlačítko Back
F2, PageUp	Tlačítko Menu
Alt+Enter	fullscreen

HALLO WORLD ☺

```
package com.example.helloandroid;

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

   public class HelloAndroid extends Activity {

      @Override
      public void onCreate(Bundle  savedInstanceState) {

         super.onCreate(savedInstanceState);
         TextView tv = new TextView(this);
         tv.setText("Hello, Android");
         setContentView(tv);
      }
}
```

HALLO WORLD PO ANDROIDSKU ☺

```
package com.example.helloandroid;

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

   public class HelloAndroid extends Activity {
       @Override
       public void onCreate(Bundle savedInstanceState) {

           super.onCreate(savedInstanceState);
           setContentView(R.layout.main);
       }
   }
```

Automaticky generovaná R třída

```
package com.example.helloandroid;

    public final class R {
        public static final class attr {
        }
        public static final class drawable {
            public static final int icon=0x7f020000;
        }
        public static final class id {
            public static final int textview=0x7f050000;
        }
        public static final class layout {
            public static final int main=0x7f030000;
        }
        public static final class string {
            public static final int app_name=0x7f040001;
            public static final int hello=0x7f040000;
        }
    }
```

Android v kostce

- Linuxové jádro řady 2.6
  - Abstrakce - odděluje HW a zbytek SW
  - Správa procesů, paměti, síťování, bezpečnost
- Dalvik virtul machine
  - Syntaxe Javy, ale pozor není kompatibilní s J2ME
  - Může běžet více virtuálních strojů současně
  - Formát .dex (Dalvik executable)

- Každá Android aplikace
  - samostatný proces
  - vlastní Dalvik virtual machine
  - unikátní Linux userID

Android v kostce

- Linuxové jádro
  - Jazyk C
  - Není standardní glibc knihovna, ale libc odvozená od BSD
- Uživatelské aplikace
  - V Javě
    (lze i nativní kód ale obvykle víc starostí než přínos)
  - Pouze syntax jazyka Java
  - Odlišnosti oproti J2SE i J2ME

  - Distribuce aplikací – Android package **.apk**

Android - bezpečnost

- iPhone 1 zdroj aplikací x Android více možností (různé přístupy)

- uživatel při instalaci aplikace **musí schválit práva** (co aplikace vyžaduje) – otázkou je, zda tak nečiní formálně

- http://mobil.idnes.cz/aplikace.asp?c=A100810_193609_programy_vok
- http://mobil.idnes.cz/aplikace.asp?c=A100818_205228_tech-a-trendy_ram
- http://www.viry.cz/go.php?p=viry&t=novinka&id=2681 (vzdálená instalace aplikací)

Linuxové jádro

- verze 2.6.*
- základní systémové služby:
  - bezpečnost
  - správa paměti
  - správa procesů
  - síťování (network stack)
  - ovladače (driver model)
- oddělení HW a SW

APLIKACE - OCHRANA

- psány v Javě
- kompilovány do Android Package (.apk)
- 1 .apk – 1 aplikace
- každá aplikace vlastní security box
  - každá – odlišný uživatel (Linux user ID)
  - práva na všechny soubory aplikace jen tomuto uživateli
  - každý proces má vlastní virtuální stroj (izolace)
  - každý aplikace běží ve vlastním procesu

Aplikace - sdílení dat

- 2 aplikace sdílet stejné UserID => přistupovat ke stejným datům, lze i ve stejném procesu, sdílet stejnou VM

- aplikace může vyžádat přístup ke kontaktům uživatele, SMS zprávám, SD kartě, kameře, Bluetooth aj.
  všechna práva musí udělit uživatel **v době instalace** aplikace

Nativní knihovny

C/C++ open source knihovny, mimo jiné:

- **Apache Harmony** – implementace Javy
- **Webkit**
  – web rendering engine (používá Safari, Chrome)
- **SQLite** – SQL databáze
- **OpenGL** – 3D grafika
- **OpenSSL**

přepsaná verze standardní C knihovny (Bionic)

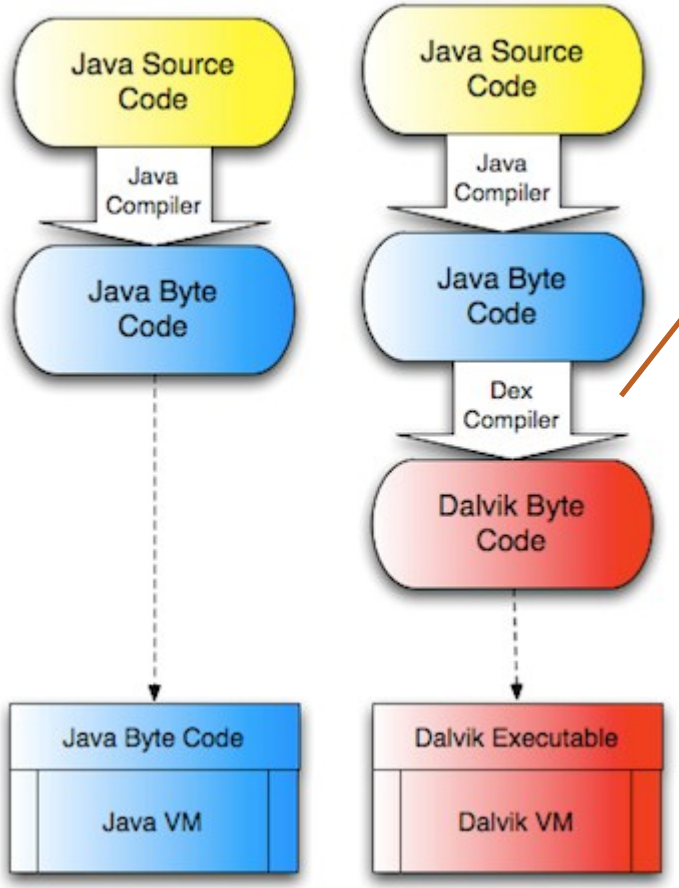
DALVIK

- VM pro Android
- bere v úvahu omezení
  (baterie, paměť, výkon CPU)

- jsme zvyklí:
  Java Source Code => Java Byte Code
- Android:
  Java Source Code => Java Byte Code
  => Dalvik Byte Code

nástroj dex: *.class -> *.dex

Překlad kódu

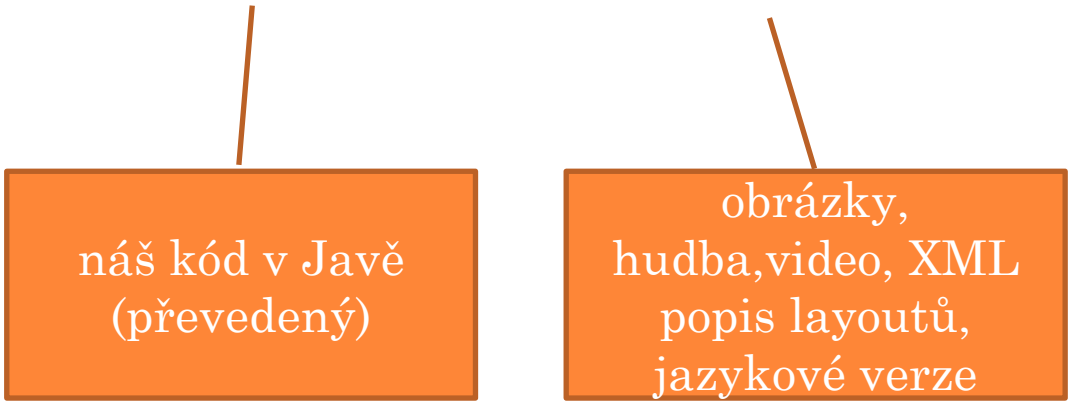


dex compiler využívá byte code, nikoliv java source

zdroj: http://ofps.oreilly.com/titles/9781449390501/ch02.html

Aplikace

- předinstalované
- stažené z Android marketů
- vámi vytvořené ☺
- **.apk** (application package file)
  - dalvik executable + resources + native code



Podpisování aplikací

- aplikace **musí** být podepsány před instalací na zařízení
- pro **vývoj** – podepsané **debug key**
  (pozor, může vypršet ☺, viz dále)
- pro **distribuci** – podepsané vlastním privátním klíčem,
  nejde debug klíčem
- vypršení certifikátu se testuje jen v **době instalace**
- update aplikace – podepsat **stejným** certifikátem
  (pokud nesouhlasí, chce jiný *package name*
  – jako úplně novou aplikaci)

EXPIRACE DEBUG CERTIFIKÁTU

- vyprší 365 dní po vytvoření

debug:
[echo] Packaging bin/samples-debug.apk, and signing it with a debug key...
*[exec] Debug Certificate **expired** on 8/4/08 3:43 PM*

- řešení:
  smazat
  c:\users\jmeno\.android\ debug.keystore
  znovu se vytvoří

- *více info ohledně certifikace aplikací viz*
  http://developer.android.com/guide/publishing/app-signing.html

Verzování aplikace

- aplikace musí být verzovaná
- součástí manifestu aplikace, 2 atributy

- android:**versionCode**
  - integer číslo, nová verze aplikace vyšší číslo
  - lze ho snadno programově porovnat
- android:**versionName**
  - řetězec, vhodné pro uživatele
  - <major>.<minor>.<point>

Př. verze aplikace

```
<?xml version="1.0" encoding="utf-8"?>
   <manifest
   xmlns:android="http://schemas.android.com/apk/res/android"
         package="com.example.package.name"
         android:versionCode="2"
         android:versionName="1.1">
      <application android:icon="@drawable/icon"
   android:label="@string/app_name">
            ...
      </application>
   </manifest>
```

Verze PLATFORMY – API Level

- velmi rychlý vývoj verzí
- jednoduchá identifikace (celé číslo int)

Android platforma poskytuje framwork API:

- Základní množina balíků a tříd
- Množina XML elementů a atributů
  - Pro deklarování manifestu
  - Pro deklarování a přístup ke zdrojům
- Množina intentů
- Množina práv, které aplikace může žádat

API LEVEL

Další informace:
http://developer.android.com/guide/appendix/api-levels.html

verze	název	API level
Android 4.0	Ice Cream Sandwitch	14
Android 3.2	Honeycomb mr2	13
Android 3.1x	Honeycomb mr1	12
Android 3.0	Honeycomb	11
Android 2.3.3	Gingerbread	10
Android 2.3	Gingerbread	9
Android 2.2	Froyo	8
Android 2.1	Eclair	7
Android 2.0.1	Eclair	6
Android 2.0	Eclair	5
Android 1.6	Donut	4
Android 1.5	Cupcake	3

Sochy před sídle google

Použití API level v manifestu

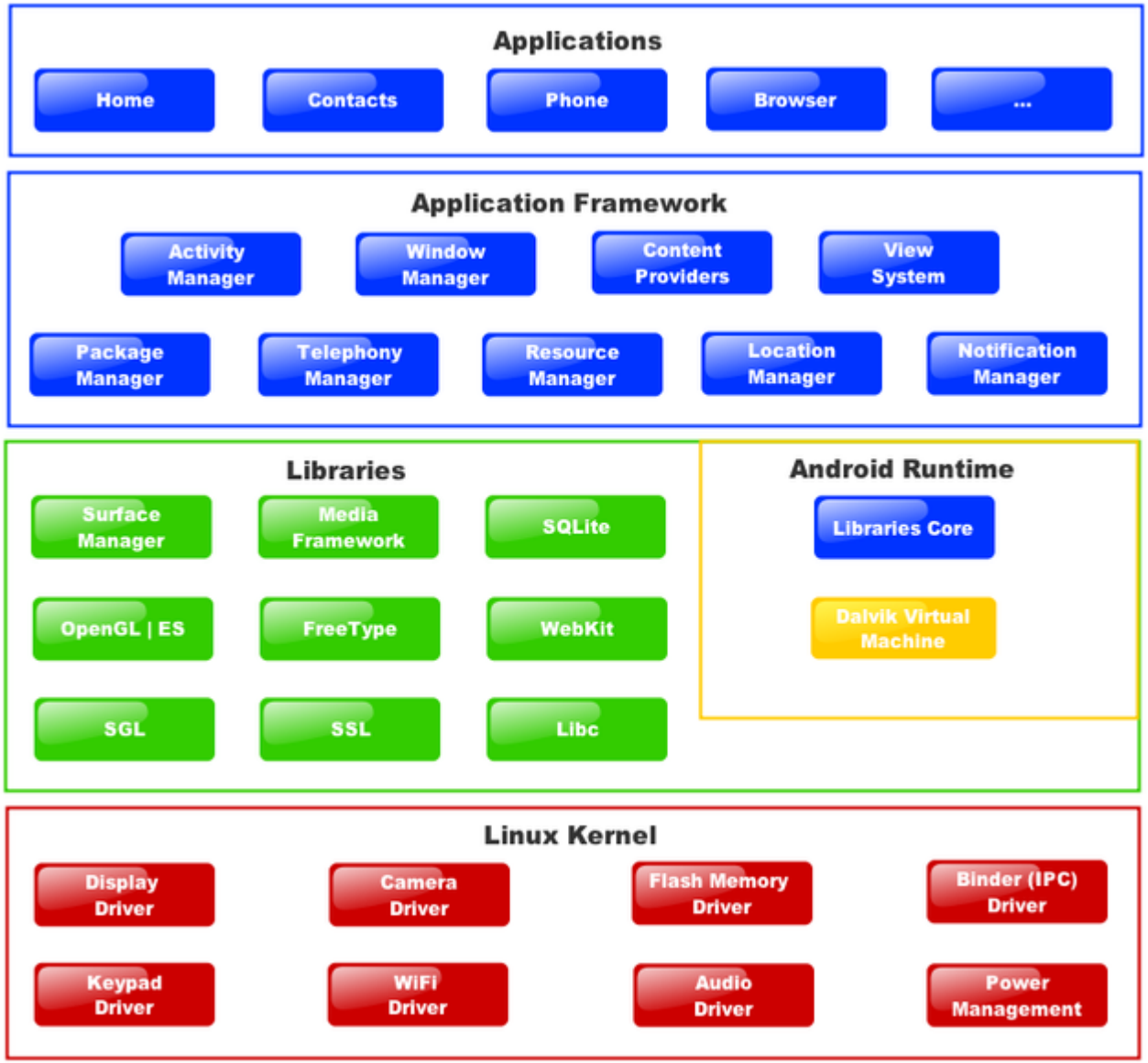
- Element v manifestu **<uses-sdk>**
- Tři základní atributy:

Použití i pro filtrování v Android Marketu

- android:**minSdkVersion**
  - Minimální API, kterou aplikace vyžaduje
- android:**targetSdkVersion**
  - API pro které je aplikace navržena, testována
- android:**maxSdkVersion**
  - Max. API na kterém je aplikace schopna běžet
  - Nedoporučuje se příliš využívat
    v novějších verzích není vynucováno

Př. Android 2.3.3 platform

- API level: *10*
- přidává např:
  NFC, Bluetooth non-secure socket connection
- vestavěné aplikace:
  - Browser, Calculator, Camera, Clock, Contacts
  - Custom Locale, Dev Tools, Downloads, Email
  - Gallery, Messaging, Music
  - IME Japonstina, Cinstina, Latin
  - Phone, Search, Settings, Speech Recorder
  - Spare Parts (developer app)

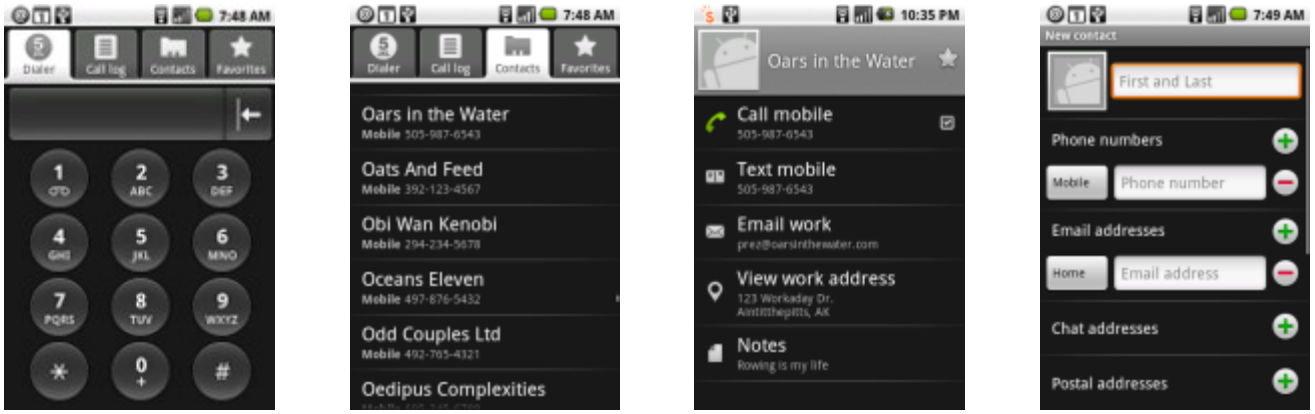

Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
Libc
Android Runtime
Libraries Core
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Driver
Power Management

Základní stavební bloky

- **Activity**
  - jedna obrazovka UI, nezávislé na sobě
  - můžeme pustit aktivitu jiné aplikace
- **Service**
  - běží na pozadí, nemá UI
    př. přehrávání hudby, stahování dat na pozadí
- **Content Provider**
  - správa sdílených dat (fs, databáze, web,…)
  - aplikace se může dotazovat na data nebo je měnit
  - jak sdílet data ven z aplikace (sms, kontakty,…)
- **Broadcast Receiver**
  - reaguje na zprávy vně i zevnitř aplikace

ACTIVITY

- Hlavní stavební blok aplikací
- aplikace se může skládat z 1 nebo několika aktivit
- Představuje vizuální UI, např. výběr položky z menu, zobrazení fotky atd.



Příklad 4 aktivit – dialer, contacts, view contact, new contact

Př: kalendář, HRA

- Kalendář - aktivity
  - Zobrazit den
  - Zobrazit týden
  - Zobrazit měsíc
  - Zobrazit agendu
  - Editace události
  - Editace preferencí
  - Zobrazení alertu
- Hra – aktivity
  - Hraní hry
  - Nastavení hry

Activity stack

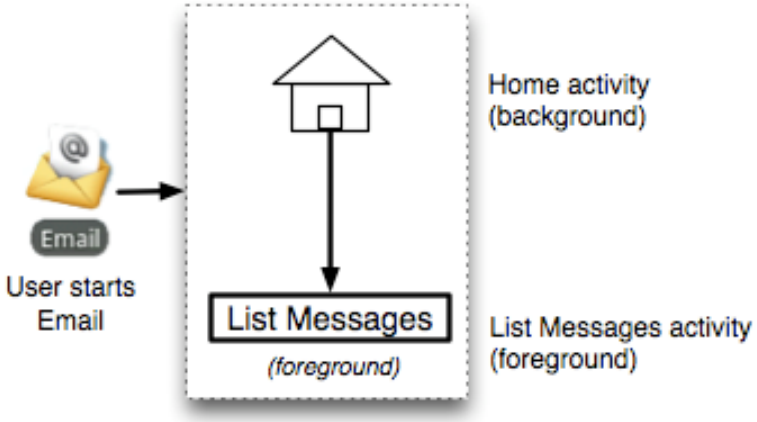
- Systém si pamatuje historii aktivit, jak se uživatel přesune k nové aktivitě
- Lineární navigační historie aktivit
- Tlačítkem BACK se lze vrátit k předchozí aktivitě

TASK

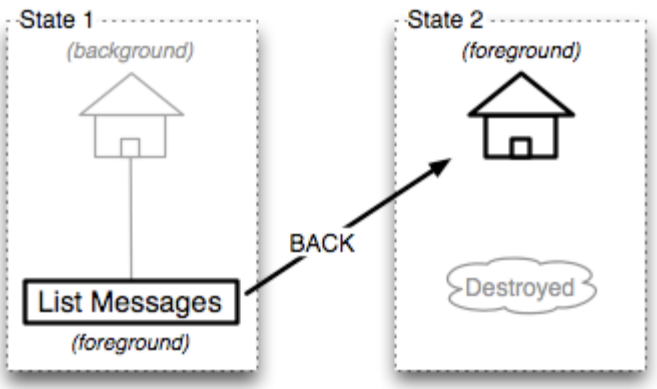
- Posloupnost aktivit, kterou uživatel vykoná pro splnění nějakého cíle

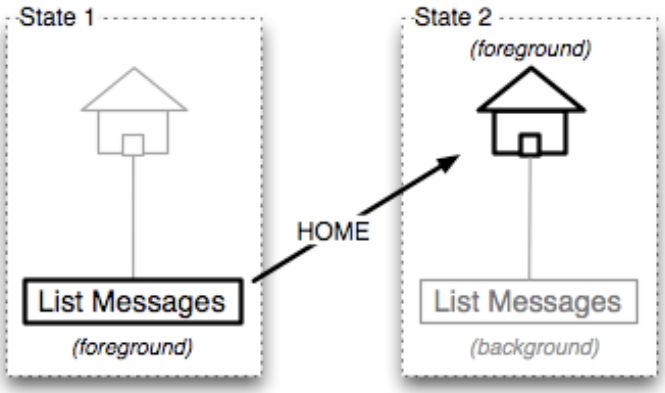
- Př.: poslání textové zprávy s přílohou

Spuštění aktivity vliv tlačítek BACK, HOME

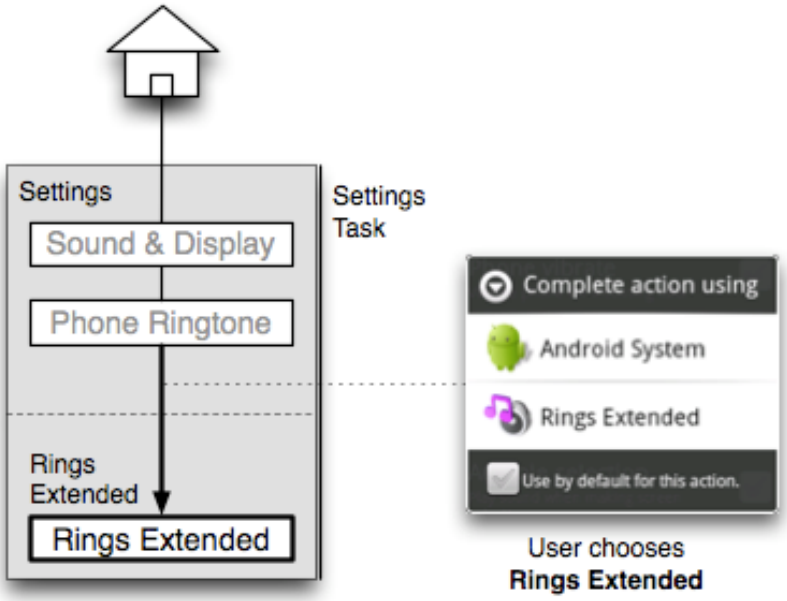


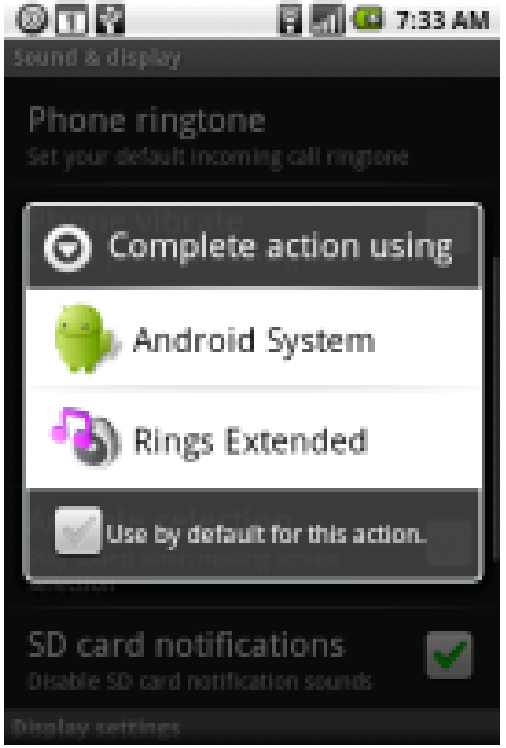
Tlačítko BACK – zrušení aktivity
Tlačítko HOME – přesun aktivity do pozadí





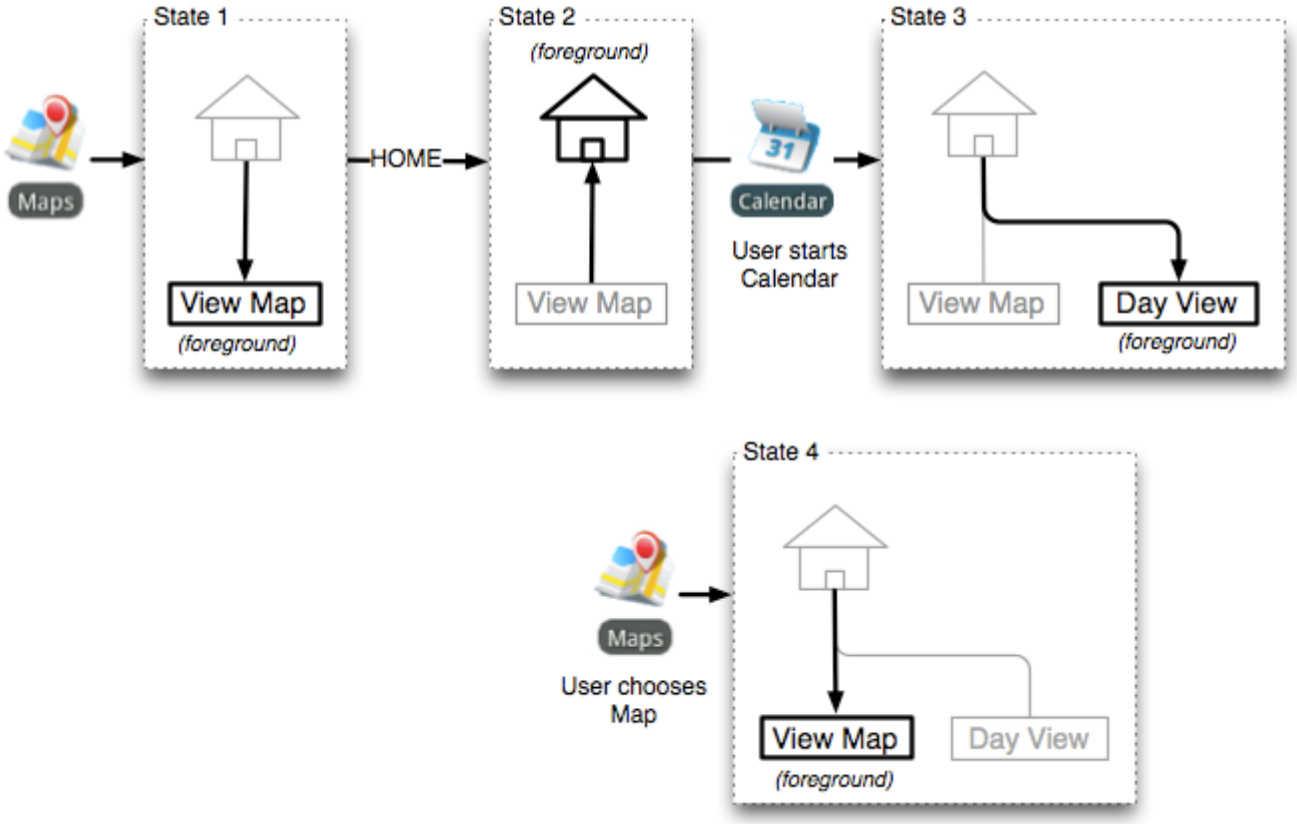
Nahrazení aktivity





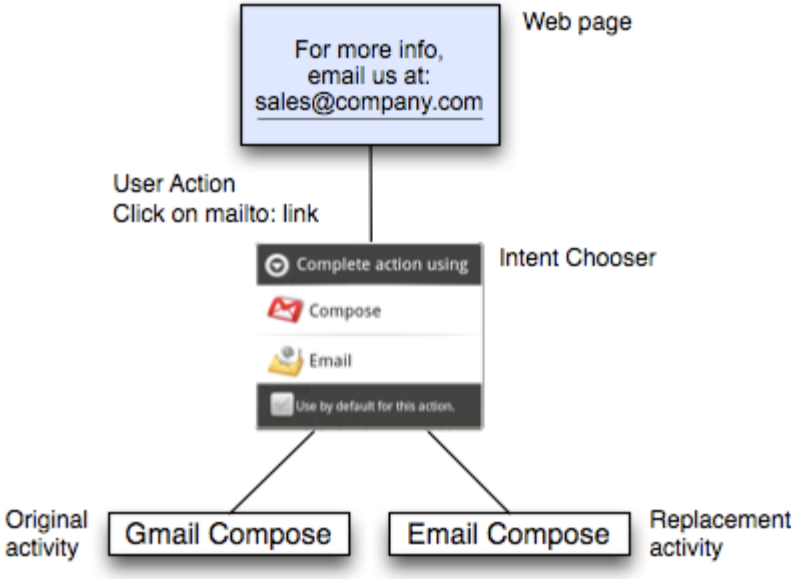
Multitasking

př.: zatímco se načítá mapa, prohlédnu si kalendář
a pak se vrátím k mapě



INTENT (ZÁMĚR)

Když uživatel vykoná určitou akci nad nějakými daty, např. touching *mailto:info@nekde.neco.cz* inicializuje se intent (intent objekt) a výsledkem může pak být spuštění aktivity



abstrakce operace, kterou je třeba provést

INTENT

- generované systémem
  - přišla SMS, změna polohy, …
- generované aplikací
  - otevři URL, vyfoť snímek, přehraj hudbu

- databáze intentů: http://www.openintents.org/en/intentstable

Vstupní body aplikace

- **tzv. LAUNCHER**, viz manifest
- seznam všech launcherů v telefonu – seznam aplikací na domovské obrazovce

- **reakce na Intent**
  - kombinace akce a např. URL
    akce:
    android.intent.action.VIEW
    android.intent.action.SEND
  - na Intent může reagovat více aplikací

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="toast.pesi.cz"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon"
   android:label="@string/app_name">
    <activity android:name=".Toastik"
            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>


</manifest>
```

Manifest

AndroidManifest.xml

- deklarace komponent
- deklarace práv, které aplikace požaduje (přístup k síti, čtení kontaktů,…)
- deklarace minimum API level
- deklarace používaného hw,sw (kamera, bluetooth služby, multitouch…)
- API library, které chce využívat (např. Google Maps library – klíč)

Manifest - komponenty

- komponenty je potřeba deklarovat:
  - <activity> .. pro aktivity
  - <service> .. pro služby
  - <receiver> .. pro broadcast receivery
  - <provider> .. pro content provider

- pokud jsou komponenty ve zdrojovém kódu, ale nejsou v manifestu – nemohou být spuštěny (výjimkou broadcast receiver)

Manifest - capability

- jaké intenty může aplikace poskytnout ostatním
- přidání elementu <intent-filter>

systém identifikuje komponenty, které mohou reagovat na intent tím, že prohledá přijatý intent s **intent filters** v manifestech aplikací

MANIFEST – POŽADAVKY APLIKACE

- např: *„chci kameru a API Level 7“*

- **screen size and density** (<supports-screens>)
- **input configuration** <uses-configuration>
  - hw klávesnice, trackball, ☺
- **device features** <uses-feature>
  - kamera, světelný senzor, bluetooth
- **platform version** <uses-sdk>

Je třeba mít na paměti rozmanitost zařízení, kde všude Android běží

MANIFEST - PERMISSIONS

- chránit kritická data a služby přes zneužitím
- každé právo jednoznačný název

android.permission.CALL_EMERGENCY_NUMBERS

android.permission.READ_OWNER_DATA

android.permission.SET_WALLPAPER

android.permission.DEVICE_POWER

aplikace deklaruje v <uses-permission>

PERMISSIONS – PŘÍKLAD

```
<uses-permission
    android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission
    android:name="android.permission.CHANGE_WIFI_STATE" />
```

seznam:

http://developer.android.com/reference/android/Manifest.permission.html

User interface - LAYOUT

- Programově
- XML soubor

```
<?xml version="1.0" encoding="utf-8"?>
   <LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/androi
   d"
               android:layout_width="fill_parent"
               android:layout_height="fill_parent"
               android:orientation="vertical" >
      <TextView android:id="@+id/text"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="Hello, I am a TextView" />
      <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
   </LinearLayout>
```

Definice UI

- Definice v layout xml souboru, unikátní id:

```
<Button android:id="@+id/my_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/my_button_text"/>
```

R.id.my_button

Pro usnadnění internacionalizace nemusíme text vkládat přímo

- Vytvoření instance s využitím layoutu:

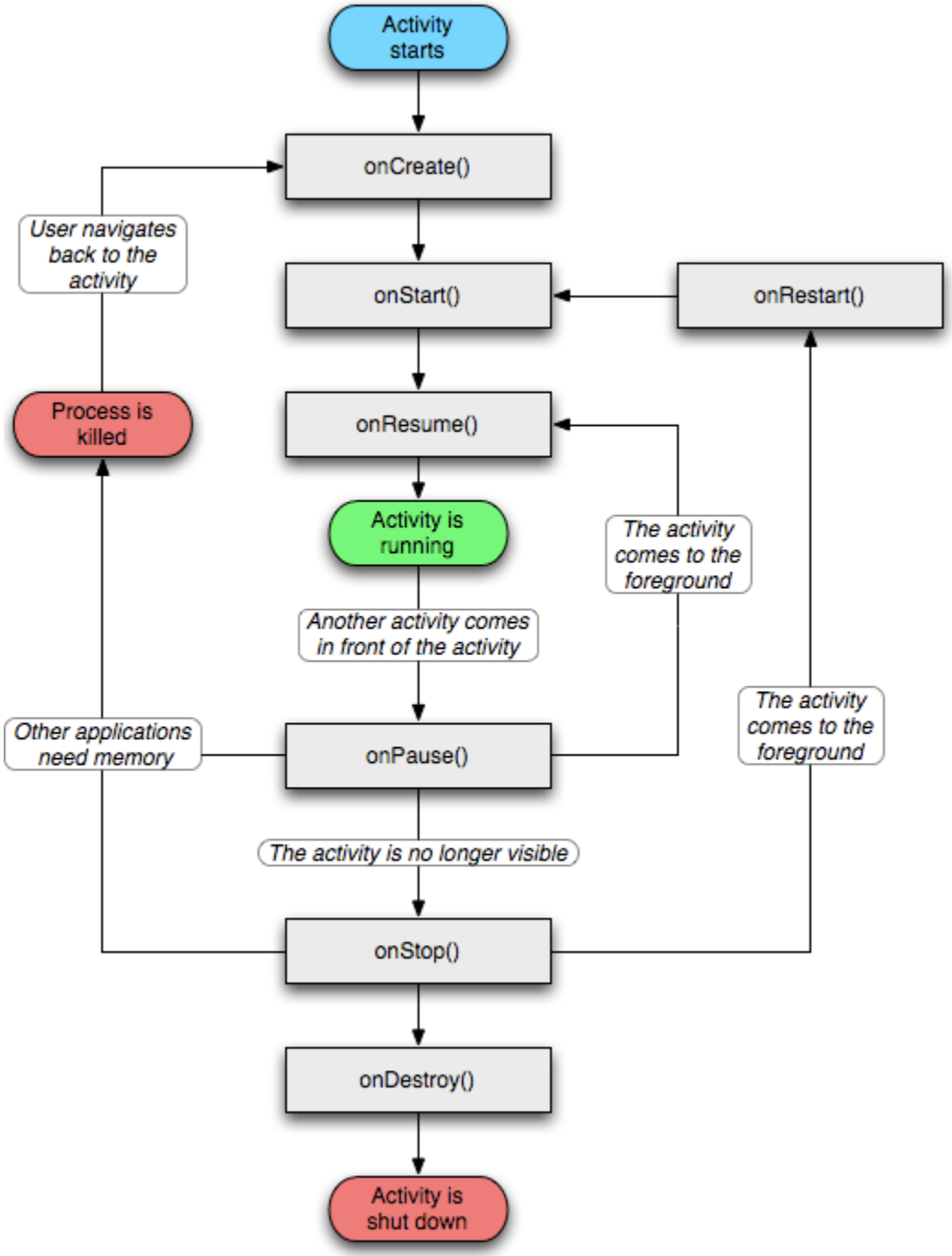
```
Button myButton = (Button)
   findViewById(R.id.my_button);
```

Životní cyklus aktivity

- Aktivní, running
  - Na popředí obrazovky, zaměřena pozornost uživatele
- Paused
  - Ztracen focus, stále viditelná (částečně) uživateli
  - Např. leží pod jinou aktivitou, která nezakrývá celou plochu
- Stopped
  - Překryta jinou aktivitou

FCE VOLANÉ PŘI PŘECHODU MEZI STAVY AKTIVITY

void onCreate(Bundle *savedInstanceState*)
void onStart()
void onRestart()
void onResume()
void onPause()
void onStop()
void onDestroy()


Activity starts
onCreate()
User navigates back to the activity
onStart()
onRestart()
Process is killed
onResume()
Activity is running
The activity comes to the foreground
Another activity comes in front of the activity
Other applications need memory
onPause()
The activity comes to the foreground
The activity is no longer visible
onStop()
onDestroy()
Activity is shut down

Obsluha události - tlačítko

Listener (posluchač)

- Programově
- Deklarativně

Každé tlačítko vlastní posluchač (typicky anonymní třída)

Lze i společný listener

Deklarace listeneru

V souboru *main.xml*:

```
android:id="@+id/button1"
    android:text="Konec"
    android:onClick="StisknutoTlacitko"
```

Deklarace události

Obslužná funkce:

```
public void StisknutoTlacitko(View v) {
   finish();
}
```

Listener programově

Do metody *onCreate()* přidáme:

```
final Button button3 = (Button)
   findViewById(R.id.button3);

button3.setOnClickListener(new
   View.OnClickListener() {
      @Override
       public void onClick(View v) {
          Toast.makeText(v.getContext(),
         "ahoj", Toast.LENGTH_LONG).show();
       }
    });
```

Použití anonymní třídy

Zobrazí info zprávu (toast)

EditText – použití

```
final EditText edit1 = (EditText)
   findViewById(R.id.editText1);

// získání číselné hodnoty
int cislo =
   Integer.parseInt(edit1.getText().toString());

// změna obsahu
edit1.setText(String.valueOf(cislo));
```

Použitá literatura

Zpracováno s využitím materiálů a obrázků z:

http://developer.android.com/

http://www.root.cz/clanky/programovani-pro-android-v-prikladech/

http://zdrojak.root.cz/clanky/vyvoj-pro-android-ii/

http://www.onlinecomputerbooks.com/free-android-books.php