Dr. Pál László, Sapientia EMTE, Csíkszereda

MOBILESZKÖZÖK ÉS ALKALMAZÁSOK
5.ELŐADÁS

2023-2024
ősz

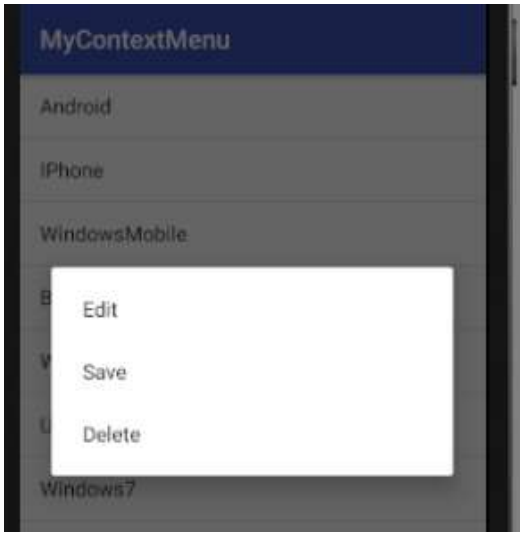
Felhasználói felület tervezése: menük
Intentek

Menü készítés

- Android alkalmazások fontos összetevői
- Három menü típus:
  - Options menu
    - A ToolBar-on (vagy ActionBar) elhelyezendő menü elemeket tartalmazó komponens
    - Az alkalmazás „globális” feladataira vonatkoznak: pld. Search, Settings, stb.
  - Példa:
    - 1.App bar
    - 2.Options menu
      - Action icons
    - 3.Overflow button
    - Option overflow menu

Menü készítés

- Context menu
  - „floating” menü: long-click esetén jelenik meg
  - Példa: listelem



- Popup menu
  - Felugró menü
  - Hasonló a context menu-re

Options menü készítés

- Főbb lépések:
  - Menü készítés erőforrás felhasználásával
    - XML állomány létrehozása a /res/menu alkönyvtárba
    - Elemek: <menu>, <item>, <group>
    - <item> elemek: *android:id*, *android:icon*, *android:title*, *android:showAsAction*
  - *onCreateOptionsMenu()* metódus felüldefiniálása, amely a menü megjelenítéséért (menu inflate) felel
  - *onOptionsItemSelected* felülírása a menüpontok lekezelésére

Options menü készítés - Példa

- XML fájl: *menu_main.xml*

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item  android:id="@+id/item1"
        android:title="Item 1"/>
    <item  android:id="@+id/item2"
        android:title="Item 2"/>
    <item  android:id="@+id/item3"
        android:title="Item 3"/>
</menu>
```

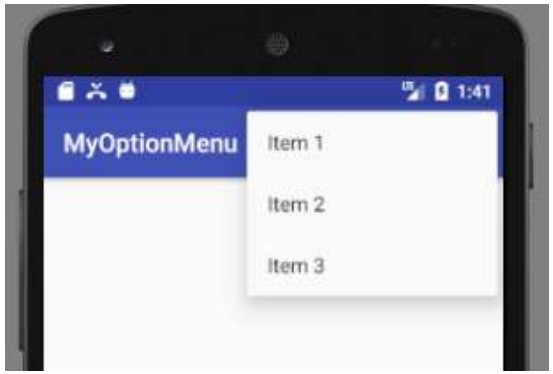
- Menü megjelenítése:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

Options menü készítés - Példa

- Menüpontok lekezelése:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.item1:
            Toast.makeText(getApplicationContext(),"Item 1 Selected",Toast.LENGTH_LONG).show();
            return true;
        case R.id.item2:
            Toast.makeText(getApplicationContext(),"Item 2 Selected",Toast.LENGTH_LONG).show();
            return true;
        case R.id.item3:
            Toast.makeText(getApplicationContext(),"Item 3 Selected",Toast.LENGTH_LONG).show();
            return true;
        default:
            return super.onOptionsItemSelected(item);
```



Options menü készítés - Példa

- Menü icon-al:
  - *android:icon*
  - *android:showAsAction*: always, ifRoom, nevers, withText

```
<item
    android:id="@+id/setup"
    android:orderInCategory="1"
    app:showAsAction="always"
    android:title="@string/setup"
    android:icon="@drawable/settings"/>

<item
    android:id="@+id/search"
    android:orderInCategory="2"
    app:showAsAction="always"
    android:title="@string/search"
    android:icon="@drawable/search"/>

<item
    android:id="@+id/email"
    android:orderInCategory="3"
    app:showAsAction="never"
    android:title="@string/email"
```

Context menü készítés

- Context menu:
  - bizonyos UI elemekhez rendelt akció (long-click) eredményeképpen jelenik meg (mint egy dialógus ablak)
  - Többnyire ListView és GridView esetén használjuk
- Létrehozás lépései:
  - View elem regisztrálása a *registerForContextView()* metódussal
  - *onCreateContextMenu()* implementálása
  - *onContextItemSelected()* implementálása

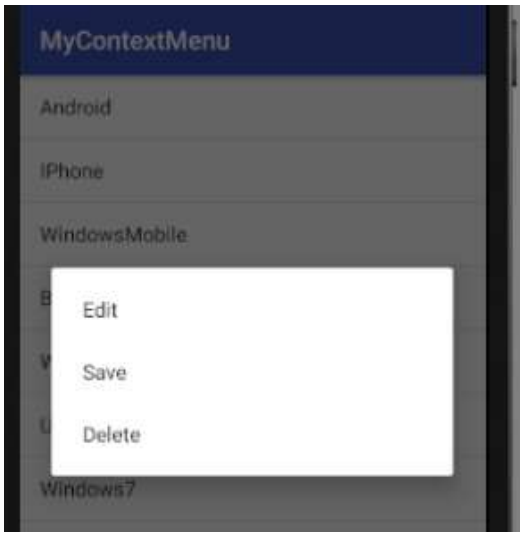
Context menü készítés - Példa

- Context menü hozzárendelése ListView-hoz
  - ListView regisztrálása

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ArrayAdapter adapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, mobileArray);

    listView = (ListView) findViewById(R.id.listviewID);
    listView.setAdapter(adapter);
    registerForContextMenu(listView);
```



  - onContextItemSelected

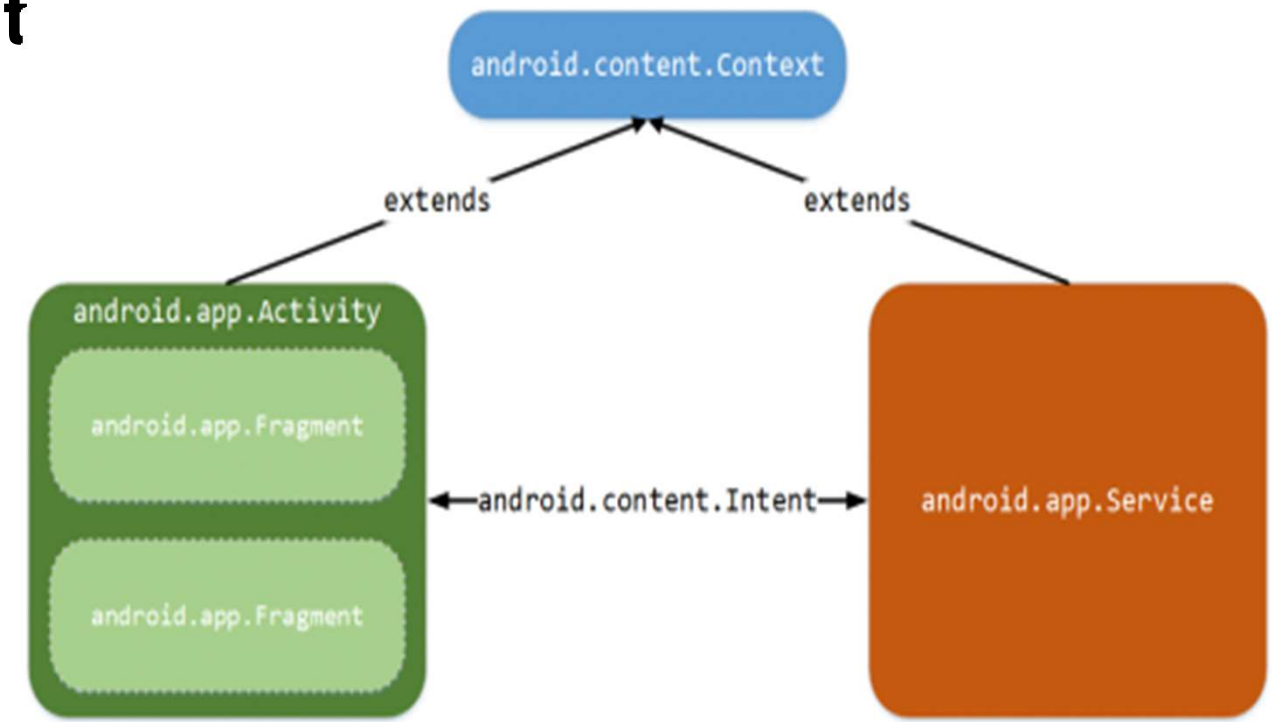
```
public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();

    switch (item.getItemId()) {
        case R.id.edit:
            Toast.makeText(this, "You have chosen the " + getResources().getString(R.string.edit) +
                    " context menu option for " + mobileArray[(int) info.id], Toast.LENGTH_SHORT).show()
            return true;
```

Intent

Alkalmazás összetevők

- **Context**
- Activity
- **Fragment**
- Service
- **Intent**



Context

- Az Android központi eleme, amely hozzáférést biztosít a rendszer különböző lehetőségeihez
  - Erőforrások elérése
  - Szolgáltatások (service)
  - Fájlok
  - Engedélyek
- Az Activity és Service osztályokban az alkalmazás Context objektuma elérhető:

  ```
  Context context = getApplication();
  ```

  vagy `this` operátor (az Activity-ben)
- Példa: `TextView TV=new TextView(this);`
- Context-el kapcsolatos tevékenységek:
  - Erőforrás betöltés, új activity indítása, nézetek (View) készítése, stb

Intent

- Alkalmazáskomponensek adatcseréjének elsődleges módszere
- Egy Java-osztály, amelyre passzív adatstruktúraként tekintünk
- Az adat, amelyet hordoz, és így maga az objektum is mindig valamilyen esemény absztrakt leírására szolgál
  - elvárt esemény (Activity vagy Service indítása)
  - bekövetkezett esemény (broadcast üzenet)

Intent

- Az *Intent* sohasem közvetlenül adódik át a komponensek között, minden esetben az operációs rendszeren keresztül történik a kézbesítése

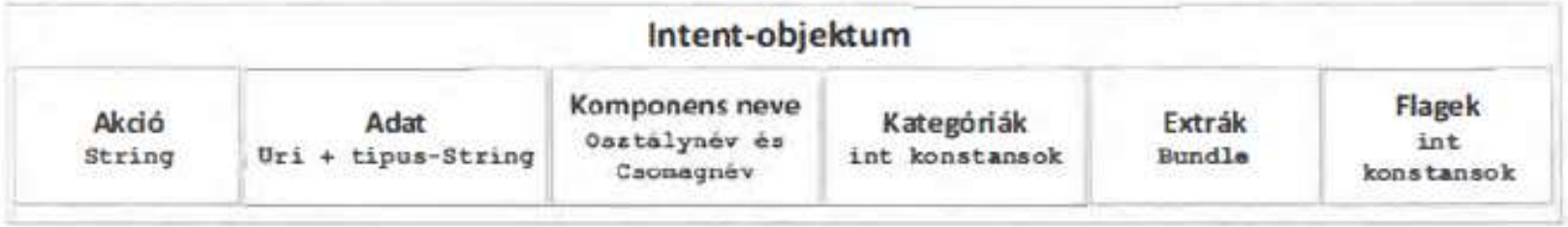
Intent

- Az Intent-objektum az Androidon a szabványos háromféle alkalmazáskomponens közti kommunikáció eszköze
- Az Intent eredete lehet tehát *Activity*, *Service*, *BroadcastReceiver* vagy maga az operációs rendszer
- A potenciális célpontok szintén ugyanezek a komponenstípusok
- Intent: a `android.content.Intent` példánya, ami átadódik a `startActivity()` vagy `startService()` metódusoknak

Intent működése

- Intent-objektum létrehozása
- Android megkeresi a megfelelő címzettet
- A címzettnek átadja neki a híváshoz tartozó és az aktiválását kiváltó adatcsomagot
- Az Intent-koncepció egyik legnagyobb erőssége:
  - a megcélzott komponens nemcsak ugyanazon a processzen belül lehet, ahonnan az Inten-tet küldték, hanem más alkalmazás valamelyik modulja is címezhető ilyen módon

Intent felépítése



- Akció (action):
  - egy sztring-konstans
  - az elvárt vagy broadcast hatására küldött Intent esetén a megtörtént eseményt jelzi
- Rengeteg előre definiált, beépített akció létezik
  - Példa:
    - ACTION_ VIEW: az Intent adatmezőjében lévő entitás megnyitása olvasásra (például fájl, névjegy)
    - ACTION_CALL: az adatmezőben átadott telefonszám felhívása
    - ACTION_BATTERY_LOW: akkufeszültség alacsony

Intent felépítése

- Adat (data): az akció meghatározhatja a mellette lévő adat típusát
- A kapcsolódó adat URI-ja és MIME-típusa szintén része az üzenetcsomagnak
  - Példa: telefonszám (tel:), kapcsolati lista (content://contacts/people), földrajzi koordináta (geo:), Internet cím (http://www.)
  - Az adatmező beállítására a setData(), a setType() és a *setDataAndType()*, míg lekérdezésére a *getData()* és a *getType()* metódusok állnak rendelkezésre

Intent felépítése

- Komponensnév(component): annak a komponensnek a neve, amelynek kezelnie kell az Intentet, az üzenet explicit címzettje
- Extrák (extra): egy *Bundle* objektum, amelyben tetszőleges mennyiségű kulcs-érték párt helyezhetünk el
  - Írás: *putExtra(String kulcs, Típus érték)*
  - Olvasás: getTípusExtra(String kulcs), ahol a „Típus” valamelyik primitív típus

Activity indítása

- Az Intent-et komponensek közti kötés megvalósítására használjuk, leggyakrabban az alkalmazás következő képernyőjére való ugrásra, azaz új Activity (vagy Service) indítására
- Példa:

```
Intentmasik Activity= new Intent(this, Masik.class);
startActivity(masikActivity);
```

- Lényeges, hogy az Activity regisztrálva legyen a manifest fájlban
- Példa: más alkalmazásban található Activity hívása

```
startActivity(new Intent("ro.cs.AnotherActivity"));
```

Activity indítása

- Paraméterek átadása:

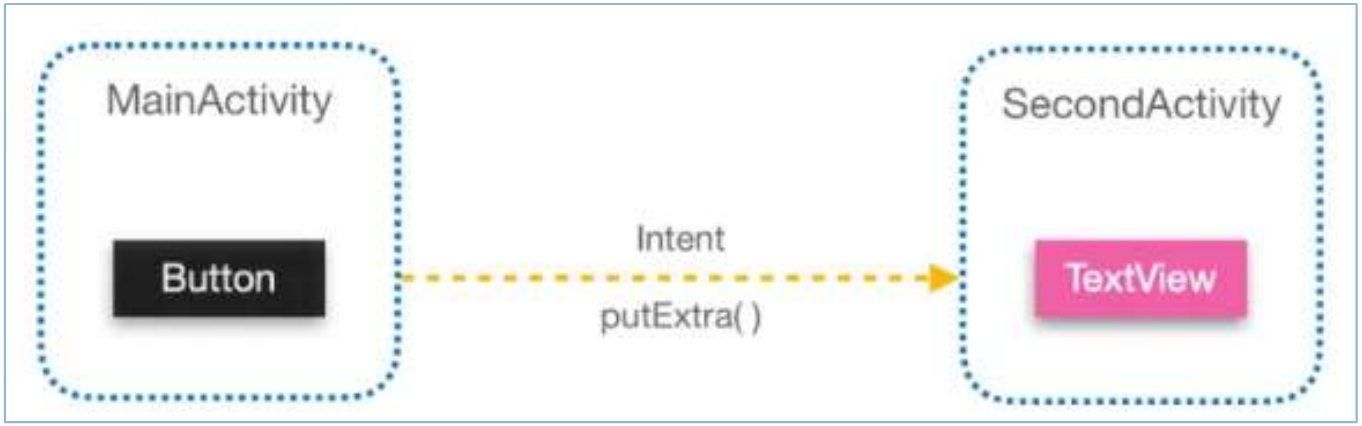
```
Intent masikActivity= new Intent(this, Masik.class);
masikActivity.putExtra("szam", 100);
masikActivity.putExtra("key","érték");
startActivity(masikActivity);
```

- Paraméterek lekérése:

```
Protected  void onCreate(BundlesavedInstanceState){
        super.onCreate(savedInstanceState);
        int szam = getIntent().getExtras().getInt("szam", 0);
        Stringkulcs = getIntent().getExtras().getString("key",
        null);
}
```

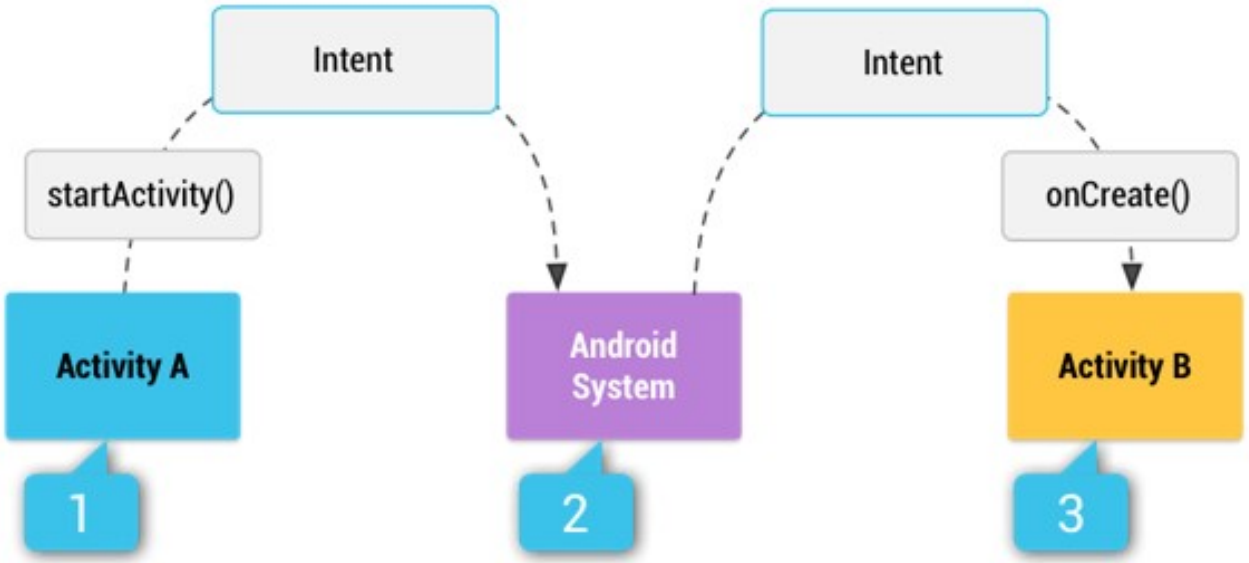
Explicit vs Implicit intent

- *Explicit Intent*:
  - a rendszer a *ComponentName* objektum osztálynév és csomagnév attribútumai alapján megkeresi az Acitivityt implementáló osztályt és az alkalmazást, amelyben szerepel, majd új példányt indít, vagy a memóriában lévőt folytatja (lásd előző példák)



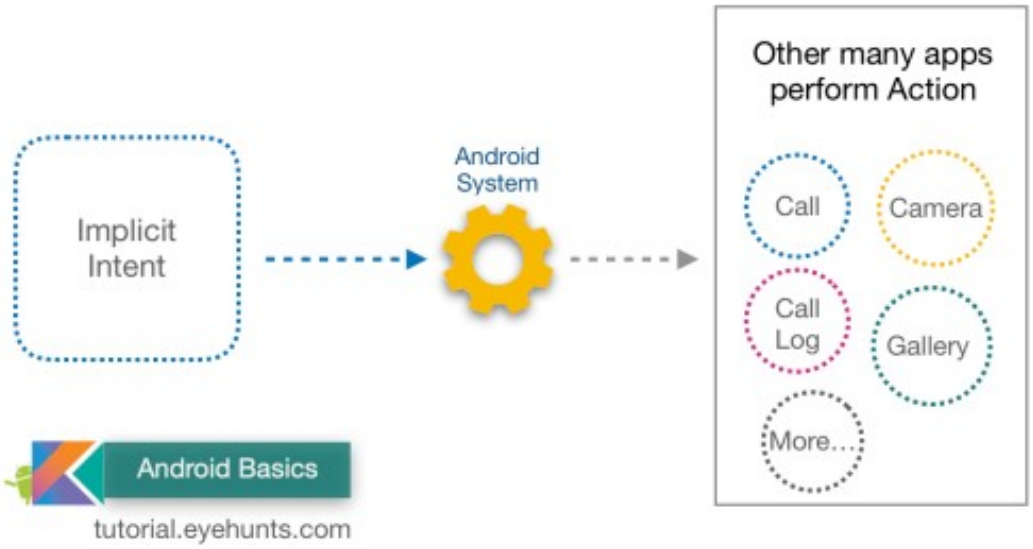
Explicit vs Implicit intent

- *Implicit Intent*:
  - az Intent *Akció* mezőjét töltjük ki, az operációs rendszerre bízzuk a megfelelő komponens kiválasztását



Explicit vs Implicit intent

- Implicit Intent:
  - Számos művelet van, amelyeket *Implicit intent* segítségével meghívhatunk:
    - *Call*, *Contact*, *Browser*, *Gallery*, *Camera*, *Alarm*, stb.



Implicit Intent példák

- Tartalom megtekintés:

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://ocw.cs.pub.ro/eim"));
```

- Tartalom keresés:

```
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
intent.setData(Uri.parse("http://www.google.ro"));
```

- Telefon hívás:

```
Intent intent = new Intent(Intent.ACTION_CALL);
intent.setData(Uri.parse("tel:0214029466"));
```

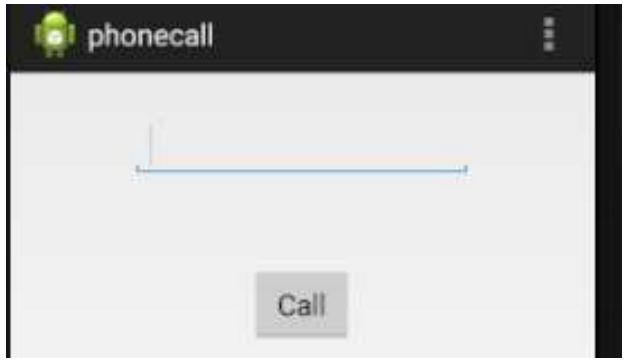
Implicit intent - Példa

- Telefonhívás kezdeményezés:
  - Hívás engedélyezés a manifest fájlban:

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

  - Hívás kezdeményezés:

```
String number=edittext1.getText().toString();
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:"+number));
startActivity(callIntent);
```



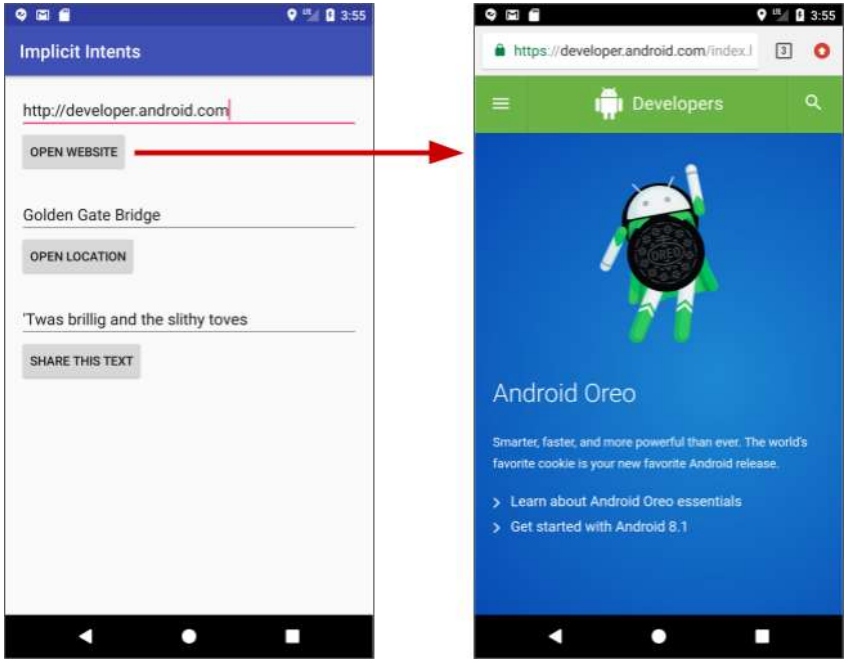
Implicit intent - Példa

- Weboldal megnyitása

```
public void openWebsite(View view) {
    // Get the URL text.
    String url = mWebsiteEditText.getText().toString();

    // Parse the URI and create the intent.
    Uri webpage = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);

    // Find an activity to hand the intent and start that activity.
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    } else {
        Log.d( tag: "ImplicitIntents", msg: "Can't handle this!");
    }
}
```



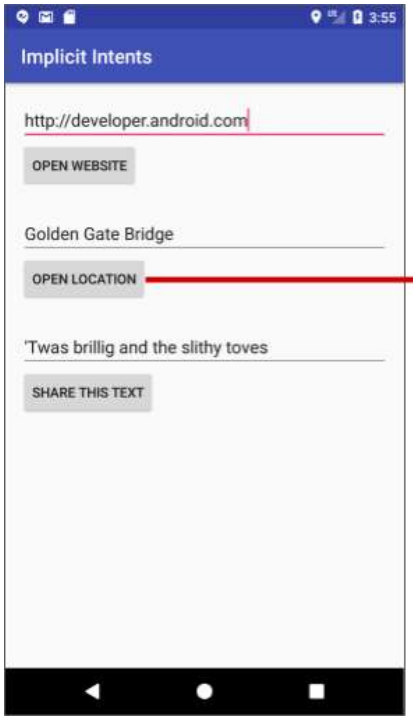
Implicit intent - Példa

- Google Map betöltése:

```
public void openLocation(View view) {
    // Get the string indicating a location. Input is not validated; it is
    // passed to the location handler intact.
    String loc = mLocationEditText.getText().toString();

    // Parse the location and create the intent.
    Uri addressUri = Uri.parse("geo:0,0?q=" + loc);
    Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);

    // Find an activity to handle the intent, and start that activity.
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    } else {
        Log.d( tag: "ImplicitIntents",  msg: "Can't handle this intent!");
    }
}
```

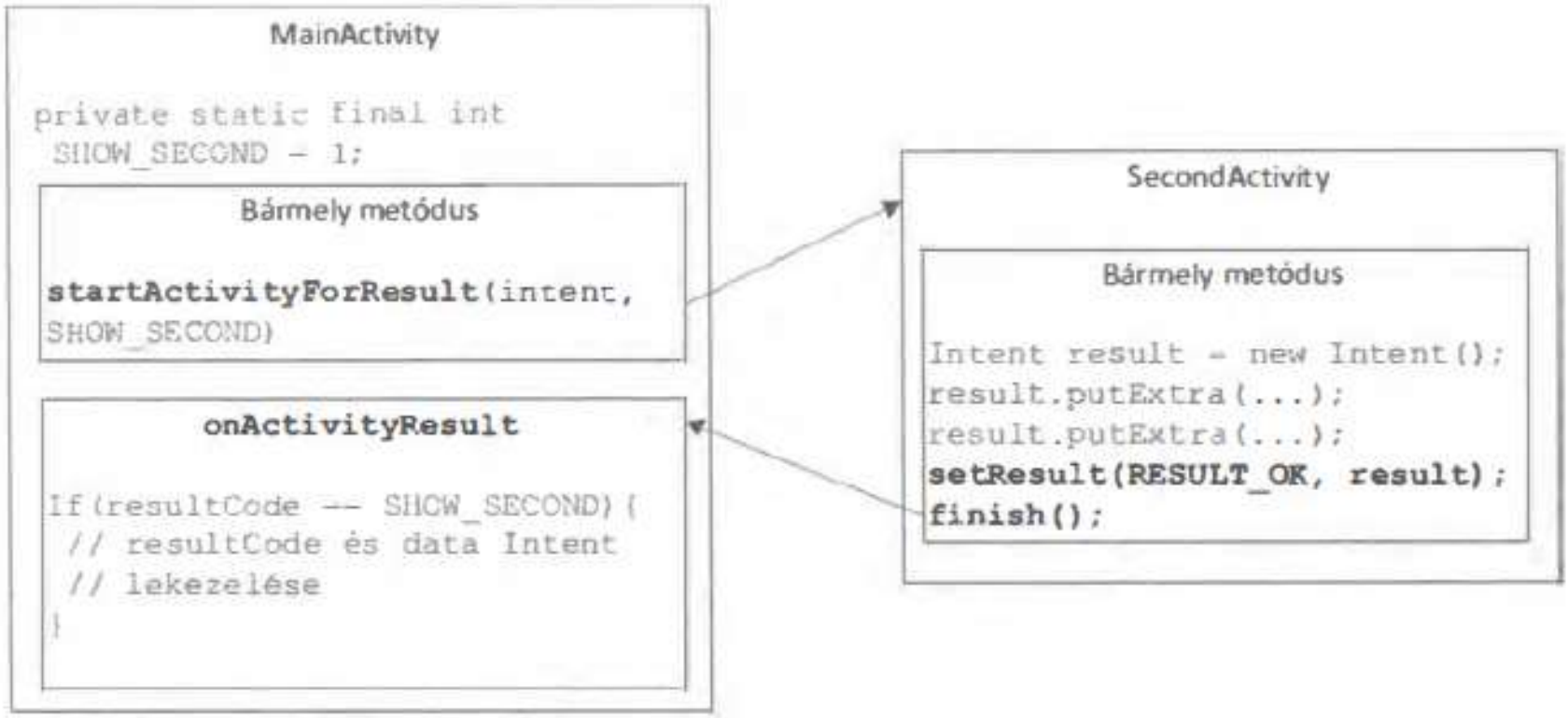


Activity visszatérési értéke

- A *startActivity* metódussal történő indításkor a hívóoldalon semmilyen visszajelzést nem kapunk arról, hogy mi történt a hívott Activityben
- Ha szeretnénk kezelni egy Activity "visszatérését", a *startActivityForResult* metódussal kell indítanunk
- Ekkor az új komponens al-Activityként jön létre a veremben, és befejeződésekor a hívóoldalon lefut egy olyan eseménykezelő, amelyben lekezelhetjük a különböző befejeződési módokat

Activity visszatérési értéke

- Befejeződésük előidézi az *onActivityResult* eseménykezelő lefutását



Activity visszatérési értéke

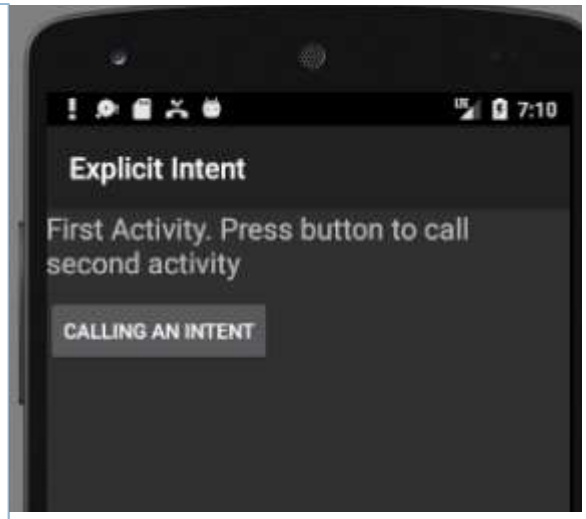
- Hívott Activityben a befejeződést előidéző finish() hívás előtt a *setResult* metódus segítségével tudjuk beállítani azokat az értékeket, amelyeket a hívó visszakap.
  - Itt lehetőségünk van megadni egy *int* visszatérési kódot, valamint opcionálisan egy egész Intent-objektumot.
- A visszatérési kódok:
  - RESULT_OK
  - RESULT_ CANCELLED
- A finish() hívás hatására az Activity befejeződik, törlődik a Back Stack tetejéről, és a vezérlés visszakerül a hívóhoz. Itt az onActivityResult() eseménykezelő hívódik meg

Explicit Intent (for result) - Példa

- ActivityOne.java

```
public void onClick(View view) {
    Intent i = new Intent(this, ActivityTwo.class);
    i.putExtra("Value1", "This value one for ActivityTwo ");
    i.putExtra("Value2", "This value two ActivityTwo");
    // Set the request code to any code you like, you can identify the
    // callback via this code
    startActivityForResult(i, REQUEST_CODE);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK && requestCode == REQUEST_CODE) {
        if (data.hasExtra("returnKey1")) {
            Toast.makeText(this, data.getExtras().getString("returnKey1"),
                    Toast.LENGTH_SHORT).show();
        }
    }
```

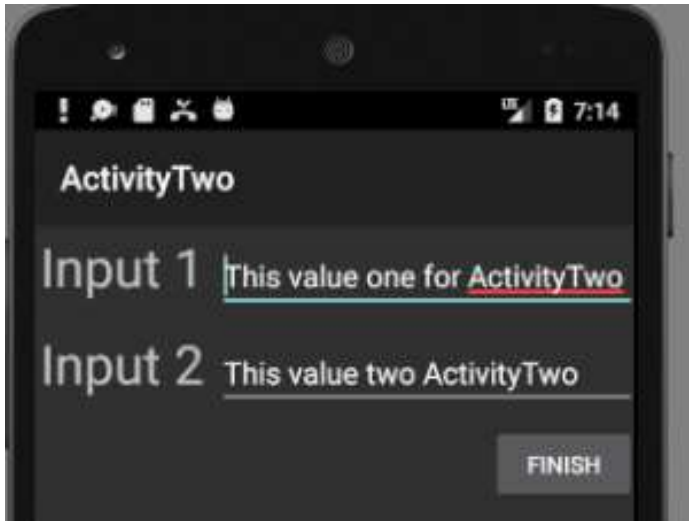


Explicit Intent (for result) - Példa

- ActivityTwo.java

```
public void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.second);
    Bundle extras = getIntent().getExtras();
    if (extras == null) {
        return;
    }
    String value1 = extras.getString("Value1");
    String value2 = extras.getString("Value2");
    if (value1 != null && value2 != null) {
        EditText text1 = (EditText) findViewById(R.id.input1);
        EditText text2 = (EditText) findViewById(R.id.input2);
        text1.setText(value1);
        text2.setText(value2);
    }
```

```
@Override
public void finish() {
    Intent data = new Intent();
    data.putExtra("returnKey1", "Swinging on a sta
    data.putExtra("returnKey2", "You could be bett
    setResult(RESULT_OK, data);
    super.finish();
}
```



Kérdések

- OptionsMenu vs. ContextMenu
- Context jellemző
- Intent működése
- Explicit vs. Implicit intent
- ActivityForResult működése

Szakirodalom

- Ekler Péter és társszerzői, Android alapú szoftverfejlesztés, Szak kiadó, 2012.
- https://codelabs.developers.google.com