Mobil- és webes szoftverek

Dr. Ekler Péter

ekler.peter@aut.bme.hu

BME AUT
Department of Automation and Applied Informatics

Miről volt szó az előző órán?

- Android e ll)
- Perziszter
  - > SQLite,
  - > SharedI
  - > File
- ContentP

Miről volt szó az előző órán?

- Android engedélyek (Permission modell)
- Perzisztens adattárolási lehetőségek
  - SQLite, ORM
  - SharedPreferences
  - File
- ContentProvider komponens

Tartalom

- Listakezelés
  - RecyclerView
  - Új elem hozzáadása
  - Törlés
  - Módosítás
- Az Intent fogalma

LISTÁK KEZELÉSE

Listák kezelése

- ListActivity / ListFragment
- Tartalmaz egy *ListView*-t, melyet a *getListView()* függvénnyel kérhetünk el
- Beállítható egy *ListAdapter*, mely a lista elemeinek tárolásáért és megjelenítésért felelős
  - Tartalmazza az elemeket (objektumok listája)
  - Felelős új elem hozzáadásáért, eléréséért és törléséért
  - A *getView(int position, View convertView, ViewGroup parent)* függvény felüldefiniálásával megadhatjuk, hogy a lista egy sorában egy elem hogyan renderelődjön
  - A convertView egy korábbi lista sor (nem tudjuk biztosan melyik), ami felhasználható az új sor létrehozására

ListAdapter példa

```
public class MyContactsAdapter extends ArrayAdapter {
    private List myContacts;
    private Context context;

    public ContactsArrayAdapter(Context context, int textViewResourceId,
            List contacts) {
        super(context, textViewResourceId, data);
        this.context = context;
        this.myContacts = contacts;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
      View v = convertView;
      if(v == null) {
        LayoutInflater inflater = LayoutInflater.from(mContext);
        v = inflater.inflate(R.layout.layout_list_item, parent);
      }
      TextView txtName = (TextView)v.findViewById(R.id.txtName);
      TextView txtMail = (TextView)v.findViewById(R.id.txtMail);
      MyContact entry = myContacts.get(position);
      txtName.setText(entry.getName());
      txtMail.setText(entry.getMail());
      return v;
    }
}
```

getView(...) használata – egyszerű eset

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
  View v = convertView;
  if(v == null) {
    LayoutInflater inflater = LayoutInflater.from(mContext);
    v = inflater.inflate(R.layout.layout_list_item, parent);
  }

  TextView txtName = (TextView)v.findViewById(R.id.txtName);
  TextView txtMail = (TextView)v.findViewById(R.id.txtMail);

  Contact entry = mList.get(position);

  txtName.setText(entry.getName());
  txtMail.setText(entry.getMail());

  return v;
}
```

getView(...) használata – ViewHolder-el (gyors!)

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
  View v = convertView;
  if(v == null) {
    LayoutInflater inflater = LayoutInflater.from(mContext);
    v = inflater.inflate(R.layout.layout_list_item, parent);
    ViewHolder holder = new ViewHolder();
    holder.txtName = (TextView)v.findViewById(R.id.txtName);
    holder.txtMail = (TextView)v.findViewById(R.id.txtMail);
    v.setTag(holder);
  }

  Contact entry = mList.get(position);
  if(entry != null) {
    ViewHolder holder = (ViewHolder)v.getTag();
    holder.txtName.setText(entry.getName());
    holder.txtMail.setText(entry.getMail());
  }
  return v;
}

static class ViewHolder {
  TextView txtName;
  TextView txtMail;
}
```

ViewHolder pattern előnyei

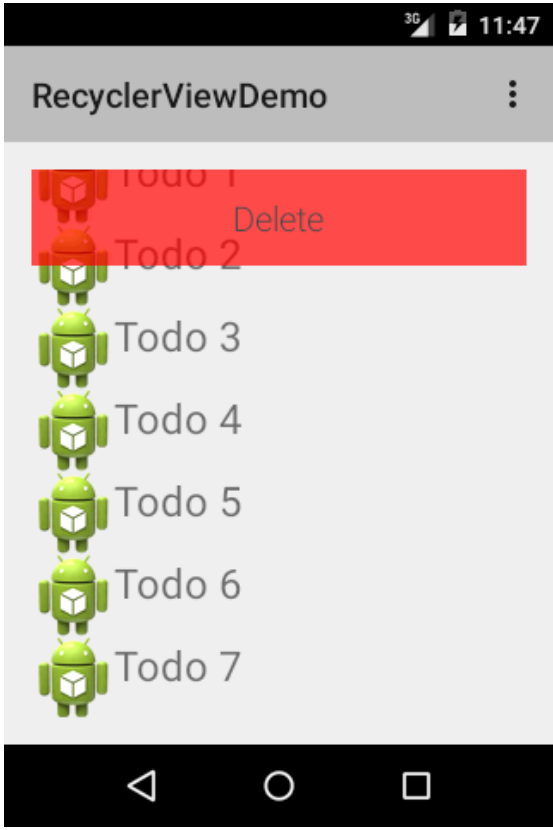
- Statikus *ViewHolder* objektum, cache támogatás
- Nincs folyamatos *findViewById(...)* hívás
- Gyors működés

RecyclerView

- A *ListView* fejlett és flexibilis változata
- *ViewHolde*r minta kikényszerítése
- Hatékony elem újrafelhasználás
- Fejlesztés:
  - *ListView*, és *Adapter* mellett egy *LayoutManager-t* is kell készíteni
- A *LayoutManager* feladata a *findViewById(...) felesleges sokszori hívása*

RecyclerView

- Fejlett ListView
- **ListView helyett javasolt**
- Flexibilis
- *ViewHolde*r pattern használata alapértelmezetten
- *Gradle dependency:*
  - compile 'com.android.support:recyclerview-v7:23.0.1'



RecyclerView.Adapter<ViewHolder> 1/3

- Inicializálás, konstruktor

```
private final List<Todo> todos = new ArrayList<Todo>();
private Context context;

public TodoAdapter(Context context) {
    this.context = context;
    for (int j = 0; j < 20; j++) {
        todos.add(new Todo("Todo " + j, false));
    }
}
```

- InitViewHolder

```
@Override
public ViewHolder onCreateViewHolder(ViewGroup
parent, int viewType) {
    View rowView =
LayoutInflater.from(parent.getContext())
            .inflate(R.layout.todo_list_row,
parent, false);

    return new ViewHolder(rowView);
}
```

ViewHolder implementáció

```
public static class ViewHolder extends RecyclerView.ViewHolder {

    private final TextView tvTodo;
    private final CheckBox cbDone;

    public ViewHolder(View itemView) {
        super(itemView);
        tvTodo = (TextView) itemView.findViewById(
                R.id.tvTodo);
        cbDone = (CheckBox) itemView.findViewById(
                R.id.cbDone);
    }
}
```

RecyclerView.Adapter<ViewHolder> 2/3

- ViewHolder binding

```
@Override
public void onBindViewHolder(ViewHolder holder, final int position) {
    final Todo todo = todos.get(position);
    holder.tvTodo.setText(todo.getTodo());
    holder.cbDone.setChecked(todo.isDone());
    holder.cbDone.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(context, "Checkbox "+position+" clicked!",
              Toast.LENGTH_SHORT).show();
        }
    });

    holder.itemView.setTag(todo);
}
```

RecyclerView.Adapter<ViewHolder> 3/3

- Elemek száma, hozzáadás, törlés

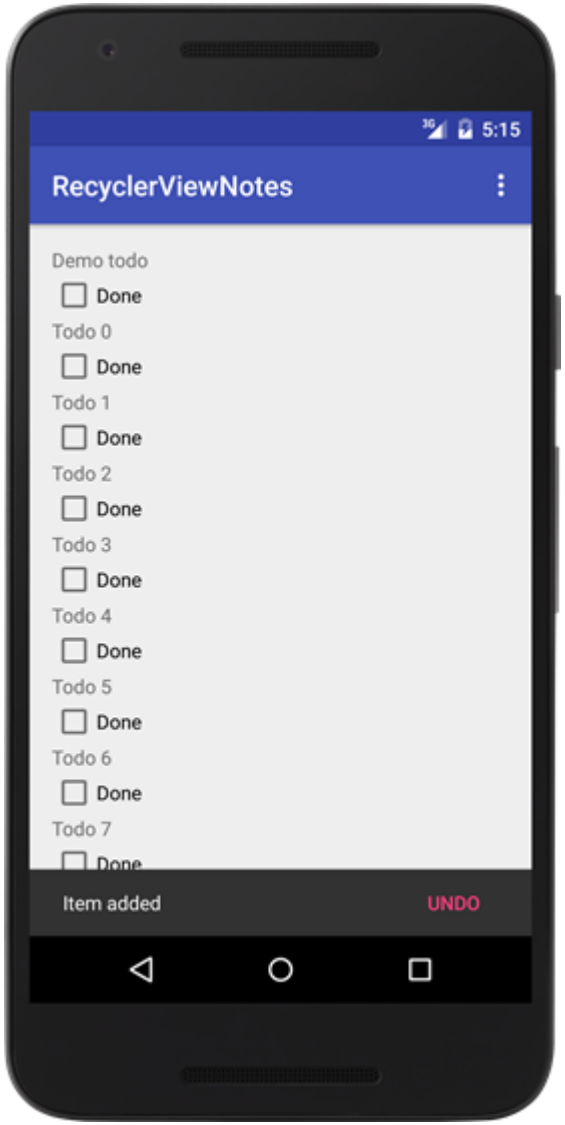
```
@Override
public int getItemCount() {
    return todos.size();
}

public void removeTodo(int position) {
    todos.remove(position);
    notifyItemRemoved(position);
}

public void addTodo(Todo todo) {
    todos.add(0, todo);
    notifyDataSetChanged();
    //notifyItemInserted(0);
}
```

Gyakoroljunk!

- Készítsünk egy Todo alkalmazást RecyclerView-val
- Jelenítsünk meg egy checkbox-ot minden elem előtt
- Valósítsunk meg hozzáadás után egy undo lehetőséget



Swipe és drag&drop gesztusok

- Távolítsuk el az elemeket swipe hatására
- Tegyük lehetővé az elemek átrendezését drag&drop-al
- *RecyclerView* támogatás:
  - `ItemTouchHelper.Callback`

ItemTouchHelper.Callback 1/2

- *isLongPressDragEnabled():*
  - True visszatérés ha a drag&drop támogatott
- *isItemViewSwipeEnabled():*
  - True visszatérésé ha a swipe támogatott
- *onMove(...):*
  - Elem mozgatás esetén hívódik meg
- *onSwipe(...):*
  - Swipe esetén hívódik meg

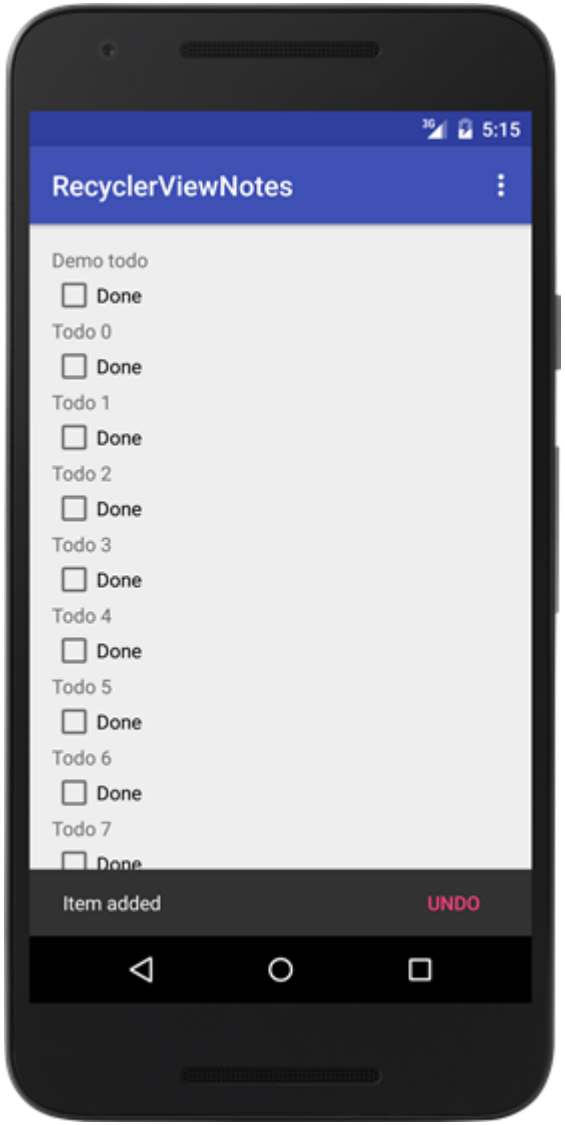
ItemTouchHelper.Callback 2/2

- Drag és swipe irányok beállítása:

```
@Override
public int getMovementFlags(RecyclerView recyclerView,
  RecyclerView.ViewHolder viewHolder) {
    int dragFlags = ItemTouchHelper.UP | ItemTouchHelper.DOWN;
    int swipeFlags = ItemTouchHelper.START | ItemTouchHelper.END;
    return makeMovementFlags(dragFlags, swipeFlags);
}
```

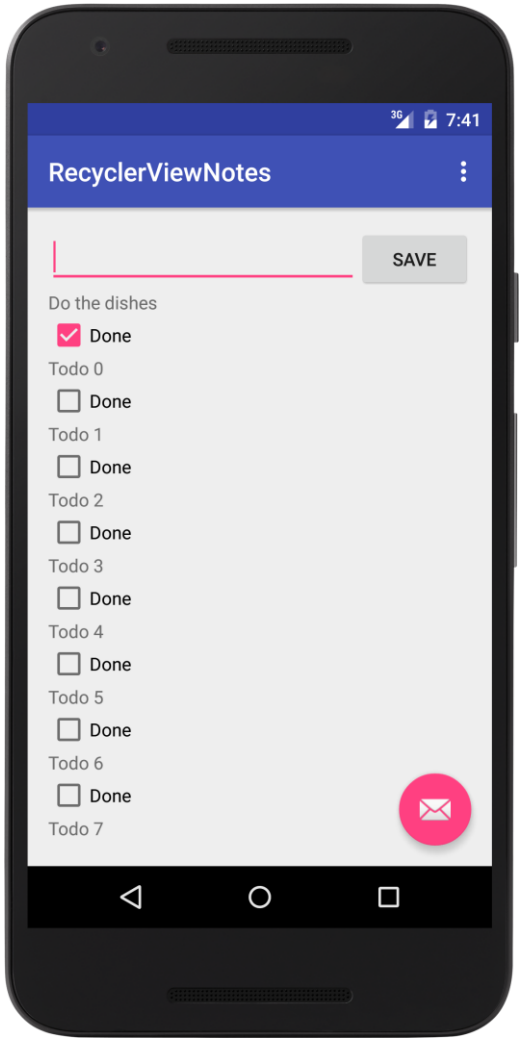
Gyakoroljunk!

- Egészítsük ki az alkalmazást swipe->törlés funkcióval
- Egészítsük ki az alkalmazást drag&drop alapú elem átrendezés funkcióval



Gyakoroljunk!

- Egészítsük ki az alkalmazást Todo hozzáadása funkcióval



Komponens közi kommunikáció, Intent

Lazán csatolt komponensek és köztük a kommunikáció

Mi a helyzet Androidon?

- Alkalmazások külön ART VM példányokban (ez is sandbox)
- Kritikus műveletekhez engedély szükséges
- Alkalmazás = komponensek halmaza

A komponensek akár alkalmazások között is kommunikálhatnak egymással (!)

- Két komponens között: ***Intent***
- Egy komponensből mindenki másnak: ***Broadcast Intent***
- Csak adat megosztása (ContentProvider)

Kommunikáció formái 1/3

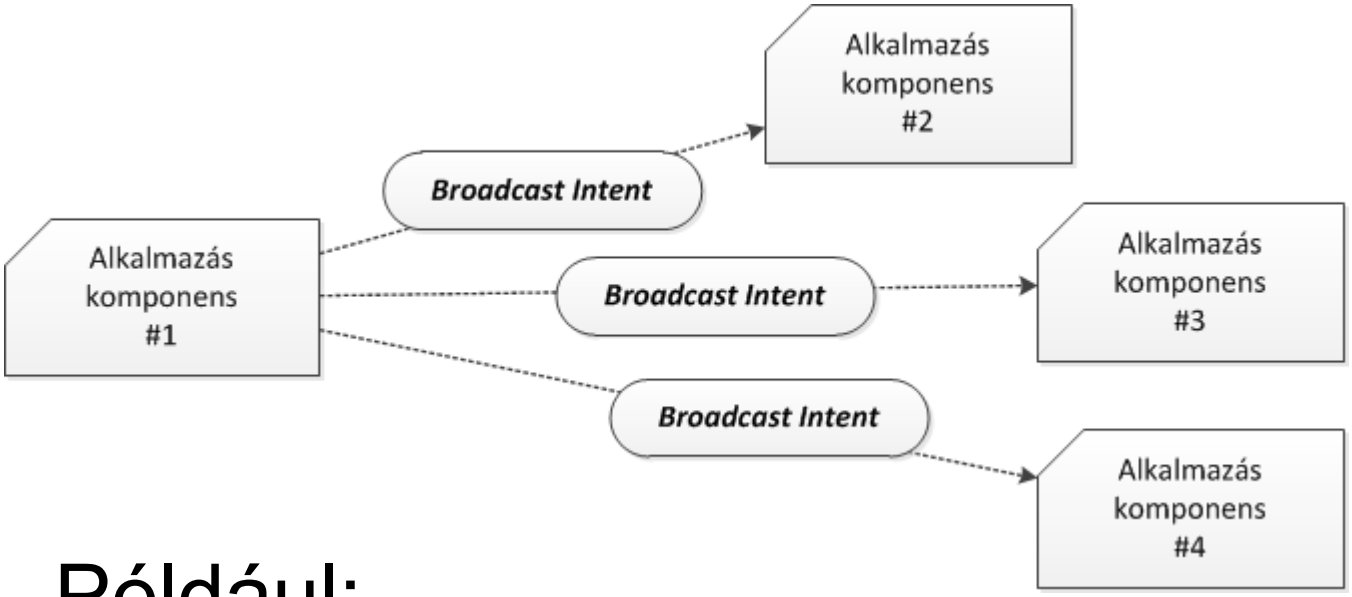
- Egyik komponensből a másikba: ***Intent***

Például:

- Következő képernyőre lépés (új Activity indítása)
- Zenelejátszó service indítása

Kommunikáció formái 2/3

- Egy komponensből mindenki másnak: ***Broadcast Intent***

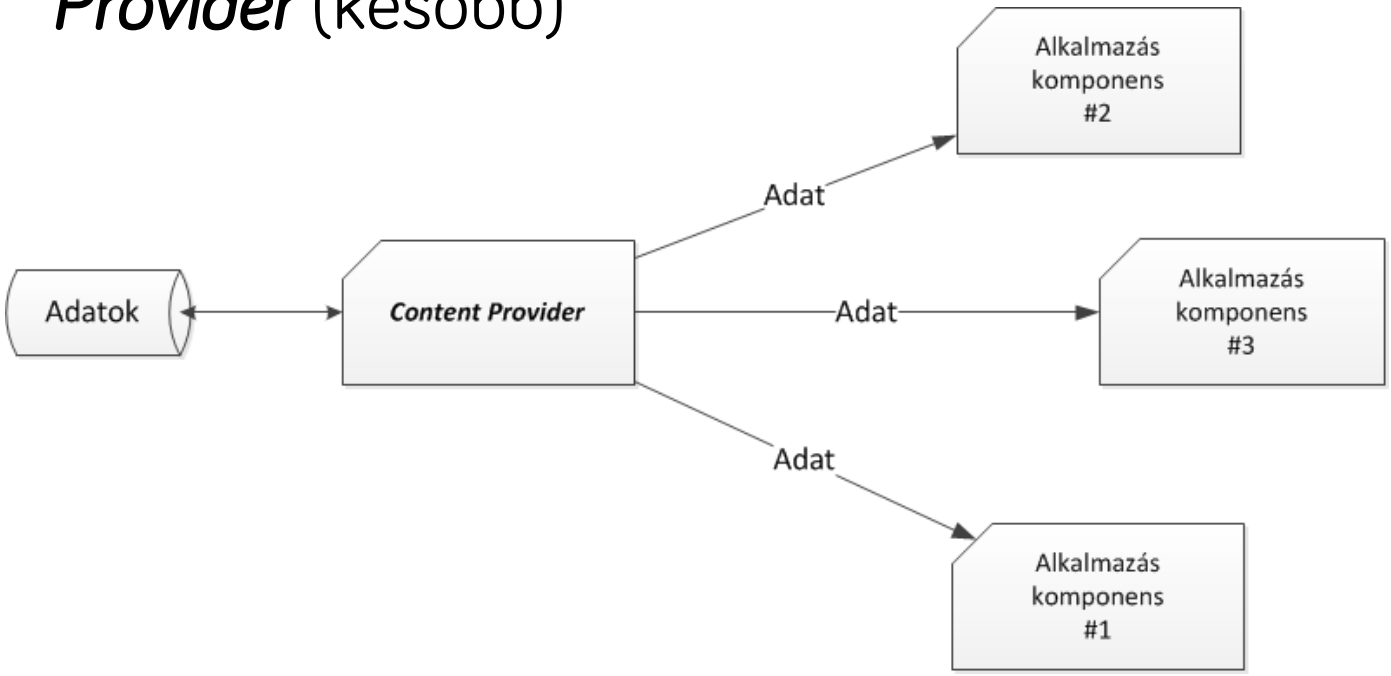


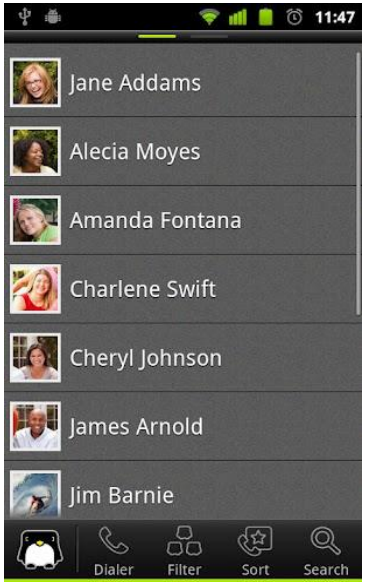
Például:

- „Akkufeszültség alacsony” rendszerüzenet

Kommunikáció formái 3/3

Adatok szolgáltatása komponensek közt: ***Content Provider*** (később)



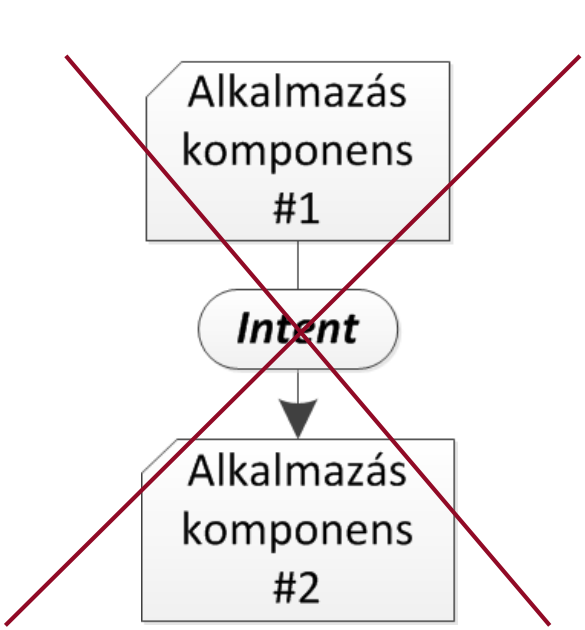


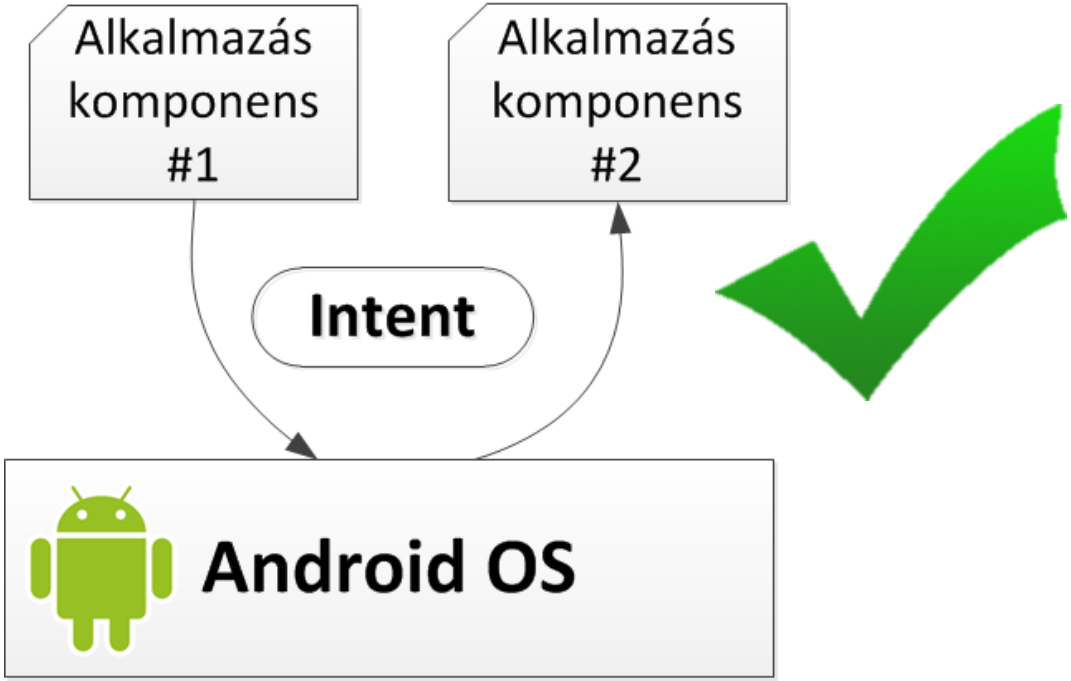
Például:

- Névjegyzék elérése saját alkalmazásból

Intent átadása

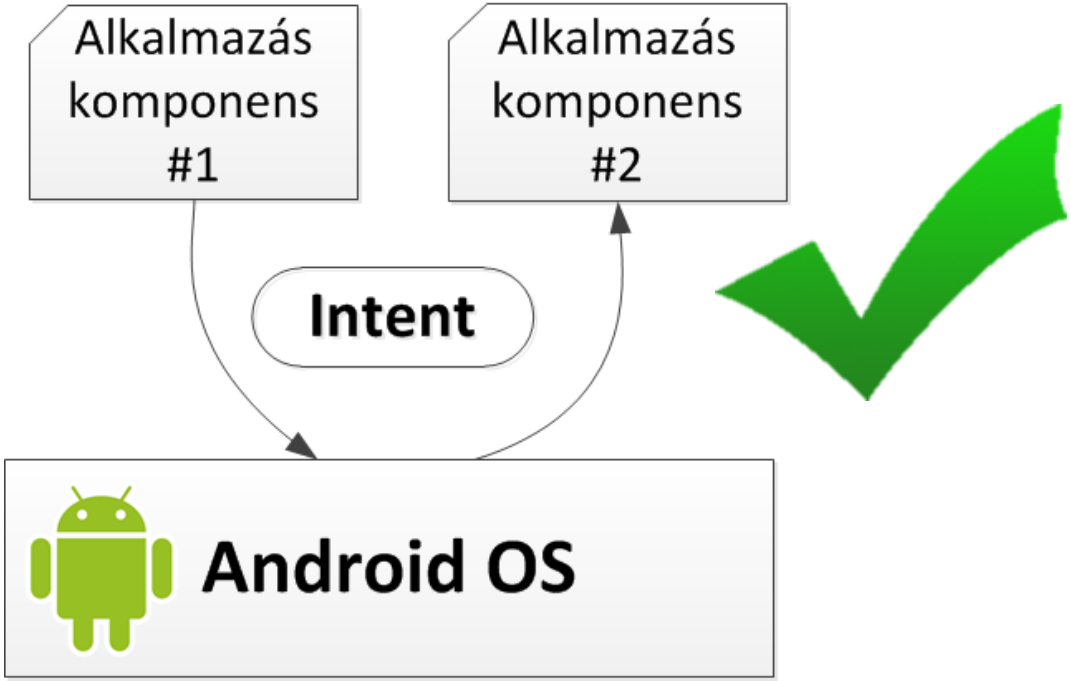
- Mindig az Android runtime-on keresztül!





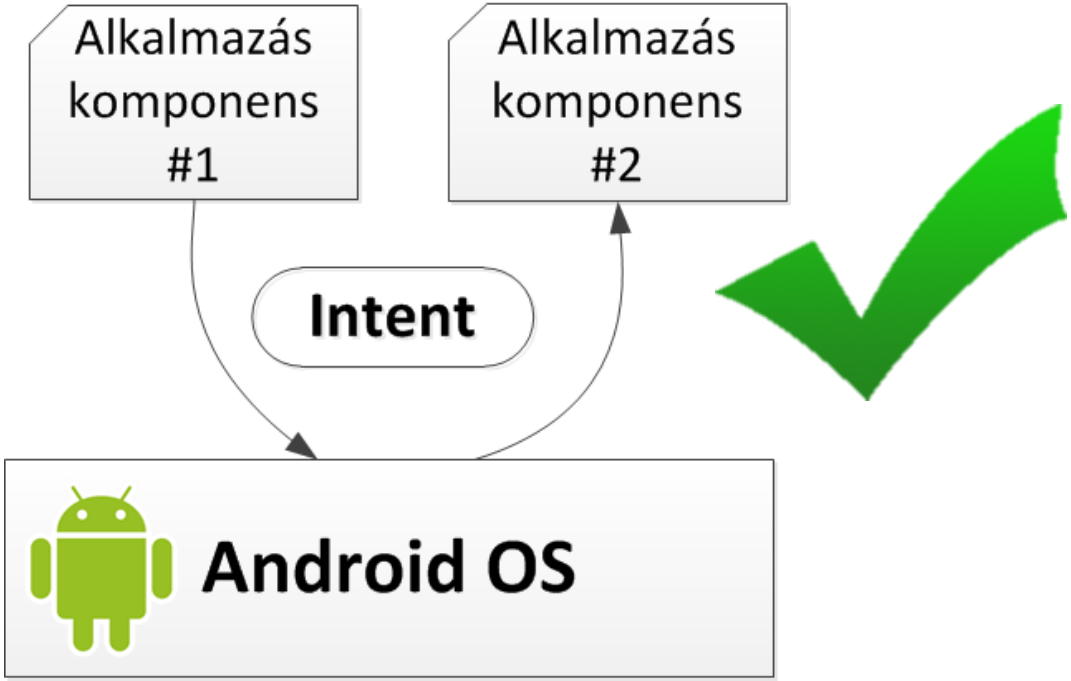
Intent átadása

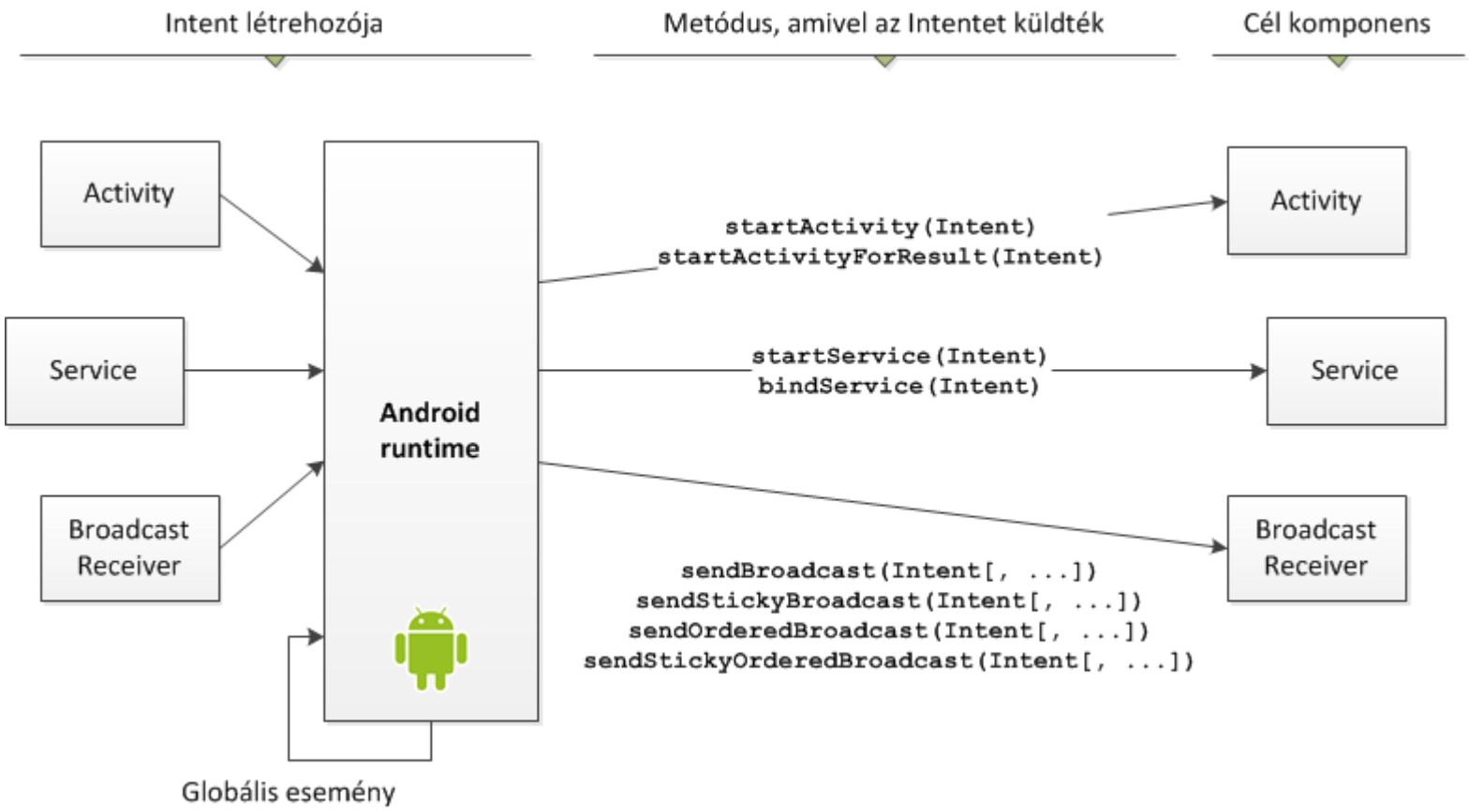
- Mindig az Android runtime-on keresztül!



Intent átadása

- Mindig az Android runtime-on keresztül!




Intent létrehozója
Metódus, amivel az Intentet küldték
Cél komponens
Activity
Service
Broadcast Receiver
Android runtime
startActivity(Intent)
startActivityForResult(Intent)
Activity
startService(Intent)
bindService(Intent)
Service
Broadcast Receiver
sendBroadcast(Intent[, ...])
sendStickyBroadcast(Intent[, ...])
sendOrderedBroadcast(Intent[, ...])
sendStickyOrderedBroadcast(Intent[, ...])
Globális esemény

Intent típusai és részei

- **Intent típusok:**
  - Explicit Intent:
    - konkrétan meg van nevezve a cél komponens
  - Implicit Intent:
    - a végrehajtandó feladat kerül leírásra
- **Intent részei:**
  - **Címzett komponens osztályneve** *(Component name)*: ha üres akkor az Android megkeresi a megfelelőt
  - **Akció** *(Action)*: az elvárt vagy megtörtént esemény
  - **Adat** *(Data)*: az adat (URI-ja és MIME típusa), amin az esemény értelmezett
  - **Kategória** *(Category)*: további kritériumok a feldolgozó komponessel kapcsolatban
  - **Extrák** *(Extras)*: saját kulcs-érték párok, amiket át akarunk adni a címzettnek
  - **Kapcsolók** *(Flags)*: Activity indításának lehetőségei

Explicit Intent

- Mindkét esetben a *startActivity()* függvényt használjuk:
  - `startActivity(Intent);`
- **Explicit hívás**: az Intent-ben kitöltjük a címzett komponens nevét (konstruktorból vagy setterrel)

```
Intent i = new Intent(getApplicationContext(),
                      ListProductsActivity.class);
startActivity(i);
```

- Ha a **`ListProductsActivity`**-ből már van példány a memóriában akkor folytatódik, ha nincs akkor az Android példányosítja és elindítja

Implicit Intent - Példa

- Telefonszám felhívása


```
Intent i = new Intent(Intent.ACTION_DIAL,
                      Uri.parse(„tel:0630-123-4567”));
startActivity(i);
```

- Névjegy kiválasztása


```
Intent i = new Intent(Intent.ACTION_PICK,
             ContactsContract.Contacts.CONTENT_URI);
startActivity(i);
```

Intent képességek

- Android alkalmazás komponensek:
  - Activity, Service, Broadcast Receiver
- Kommunikáció köztük: Intentekkel
- Nem csak alkalmazáson belül, hanem azok között is lehetséges
  - Használhatunk más alkalmazásban lévő komponenst
  - Kiajánlhatjuk a sajátunkat
- Rendszerszintű eseményeket kezelhetünk

Intent Filter

- Lehetséges a saját alkalmazásunk funkcióinak kiajánlása mások számára
  - Az Androidban beépítve vannak ilyenek, ld. Intent Action (pl. ACTION_CALL, ACTION_IMAGE_CAPTURE)
- Az AndroidManifest-ben kell deklarálni (miért?)
- Ha nincs Intent filter beállítva, akkor a komponens kizárólag explicit intentet képes fogadni
- Ha van Intent filter, akkor explicit és implicit intenteket is ki tud szolgálni

Mit nevezünk Explicit Intentnek?

A. Ami kihív az alkalmazásból.

B. Ami explicit képet ad vissza hívás után.

C. Ami konkrét telefonszámot hív fel.

D. Amikor megadjuk a konkrét osztályt (komponenst) akinek a kérést küldjük.

Összefoglalás

- RecyclerView
  - Különféle gesztusok kezelése
- Intent
  - Explicit Intent
  - Implicit intent

Kérdések