**Mi az a RecyclerView, és miért használjuk?**

Az **RecyclerView** az Android fejlesztésben az adatok hatékony megjelenítésére szolgál, például listák vagy rácsszerkezetek formájában.

1. **View Recycling (Újrafelhasználás):** Csak azokat az elemeket tölti be, amelyek éppen a képernyőn láthatók, ezzel memóriát takarít meg.
2. **LayoutManager:** Meghatározza az elemek elrendezését (például lineáris lista, rács, stb.).
3. **Adapter:** Az adatok kezelése és az elemek feltöltése az egyéni nézetekbe.
4. **ViewHolder Pattern:** Az elemek gyorsabb megjelenítésére szolgál azáltal, hogy a nézeteket gyorsítótárazza.

**RecyclerView létrehozása**

**1. Felhasználói felület**

Az XML fájlban az alábbi kóddal definiáljuk a RecyclerView-t, ez önmagában jelenti az egész görgethető listát:

<androidx.recyclerview.widget.RecyclerView

android:id="@+id/recyclerView"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="8dp"

android:background="@android:color/white"/>

**2. Adatmodell létrehozása (User.kt)**

Az adatmodell azokat az adatok reprezentálja, amelyeket meg akarunk jeleníteni. Ebből általában egy listánk lesz, ezt a listát tartalmazza a RecyclerView. Itt létrehozunk egy User osztályt, amely a következő adatokat tartalmazza:

data class User(

val name: String,

val email: String,

val profileImage: Int // Drawable resource ID

)

**3. Egyéni listaelem létrehozása (list\_item.xml)**

A listában levő elemek kinézetét ebben az XML fájlban definiáljuk, tehát ez csak 1 user megjelenítését határozza meg egy önálló xml fájl formájában:

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:padding="12dp"

android:orientation="horizontal">

<ImageView

android:id="@+id/profileImage"

android:layout\_width="50dp"

android:layout\_height="50dp"

android:src="@drawable/ic\_user"/>

<LinearLayout

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:orientation="vertical">

<TextView

android:id="@+id/nameText"

android:textSize="18sp"

android:textStyle="bold"/>

<TextView

android:id="@+id/emailText"

android:textSize="14sp"

android:textColor="@android:color/darker\_gray"/>

</LinearLayout>

</LinearLayout>

**4. Adapter létrehozása (UserAdapter.kt)**

## **1. Az Adapter osztály és a generikus típusok**

class UserAdapter(private val userList: List<User>) : RecyclerView.Adapter<UserAdapter.UserViewHolder>()

* **RecyclerView.Adapter<UserAdapter.UserViewHolder>()**  
  Ez az adapter osztály az RecyclerView.Adapter-ből származik, és egyedi UserViewHolder típusát használja.  
  Az adapter felelős az egyes nézetek létrehozásáért és az adatok megjelenítéséért.
* **private val userList: List<User>**  
  Az adapter konstruktorában átadunk egy **userList** nevű listát, amely az adatok gyűjteménye. A lista elemeit meg kell jeleníteni az egyes listanézetekben.

## **2. ViewHolder osztály**

class UserViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {

val profileImage: ImageView = itemView.findViewById(R.id.profileImage)

val nameText: TextView = itemView.findViewById(R.id.nameText)

val emailText: TextView = itemView.findViewById(R.id.emailText)

}

**Miért van szükség ViewHolder-re?**

* A ViewHolder koncepció segít **optimalizálni a lista teljesítményét**, elkerülve a felesleges hívásokat a findViewById() metódusra.
* Az újrafelhasználás miatt a nézeteket gyorsabban tudjuk be- és kirenderelni.
* **RecyclerView.ViewHolder(itemView)**: Az alap ViewHolder osztályt örökli, amely tartalmazza az adott listaelem nézetének referenciaobjektumát.

**Mezők:**

* **profileImage: ImageView** – A profilkép megjelenítésére szolgáló képnézet.
* **nameText: TextView** – A felhasználó nevének megjelenítésére szolgáló szöveges nézet.
* **emailText: TextView** – Az e-mail cím megjelenítésére szolgáló szöveges nézet.

## **3. onCreateViewHolder() – Új listaelemek létrehozása**

kotlin

MásolásSzerkesztés

override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): UserViewHolder {

val view = LayoutInflater.from(parent.context).inflate(R.layout.list\_item, parent, false)

return UserViewHolder(view)

}

**Mi történik itt?**

1. **LayoutInflater.from(parent.context)** – Az LayoutInflater felelős a list\_item.xml elrendezés átalakításáért egy megjeleníthető View objektummá.
2. **inflate(R.layout.list\_item, parent, false)**
   * A R.layout.list\_item az egyedi listaelem elrendezése, amely a listában jelenik meg.
   * A **parent** az az objektum, amely tartalmazza a RecyclerView elemeit.
   * A **false** paraméter azt jelenti, hogy a nézetet nem kell azonnal a szülőhöz csatolni.
3. **Visszatérés:** Létrehozunk egy **UserViewHolder** objektumot, amely tárolja az újonnan létrehozott nézetet.

## **4. onBindViewHolder() – Adatok megjelenítése a nézetben**

kotlin

MásolásSzerkesztés

override fun onBindViewHolder(holder: UserViewHolder, position: Int) {

val user = userList[position]

holder.nameText.text = user.name

holder.emailText.text = user.email

holder.profileImage.setImageResource(user.profileImage)

}

**Mi történik itt?**

1. **val user = userList[position]**
   * Lekérjük az aktuális pozíción lévő felhasználói objektumot a listából.
2. **Nézetek frissítése:**
   * A **holder.nameText.text = user.name** – A nézetben megjelenik a felhasználó neve.
   * A **holder.emailText.text = user.email** – Megjeleníti az e-mail címet.
   * A **holder.profileImage.setImageResource(user.profileImage)** – Beállítja a profilképet.

**Miért fontos ez a lépés?**

* Az adatok megfelelő indexhez tartozó nézethez való rendelése.
* A lista gyors frissítése és a hatékony memóriahasználat érdekében a nézetek újrahasznosítása.

## **5. getItemCount() – Az adatlista méretének visszaadása**

kotlin

MásolásSzerkesztés

override fun getItemCount() = userList.size

**Mit csinál ez a függvény?**

* Visszaadja a lista méretét, vagyis hogy hány elemet kell megjeleníteni.
* Ezt a RecyclerView automatikusan használja annak meghatározására, hogy hány elemre van szüksége.

### ****Hogyan működik az egész folyamat?****

1. **A RecyclerView meghívja az onCreateViewHolder függvényt**, amikor új elemre van szükség a képernyőn.
2. **Az onBindViewHolder beállítja az adatokat**, amikor egy elem újra felhasználásra kerül vagy első alkalommal megjelenik.
3. **A ViewHolder tárolja a nézetek referenciáit**, hogy ne kelljen minden egyes alkalommal újra keresni őket a DOM-ban.
4. **Az újrahasznosítás miatt kevesebb memóriát fogyaszt**, és jobb teljesítményt biztosít nagy adatlisták esetén.

### ****Összegzés****

* **ViewHolder**: Gyorsítótárazza az egyes listaelemek nézeteit.
* **Adapter**: Az adatok és a nézetek közötti kapcsolatot biztosítja.
* **RecyclerView működése**:
  + Nézetek létrehozása (onCreateViewHolder).
  + Adatok beállítása (onBindViewHolder).
  + Elem méretének visszaadása (getItemCount).
* **Hatékonyság**: A RecyclerView az újrafelhasználás révén jobb teljesítményt biztosít, mint a ListView.

**5. RecyclerView konfigurálása az Activity-ben (MainActivity.kt)**

A fő tevékenység (activity) inicializálja és kezeli a RecyclerView-t.

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

val recyclerView: RecyclerView = findViewById(R.id.recyclerView)

val users = listOf(

User("Kovács Péter", "peter.kovacs@email.com", R.drawable.ic\_user),

User("Nagy Anna", "anna.nagy@email.com", R.drawable.ic\_user)

)

recyclerView.layoutManager = LinearLayoutManager(this)

recyclerView.adapter = UserAdapter(users)

}

}

**Mi történik itt?**

1. **Adatok létrehozása (Statikus lista)**
2. **LayoutManager beállítása**
   * A LinearLayoutManager(this) függőleges görgetést biztosít.
3. **Adapter beállítása a RecyclerView-hoz**

**Összegzés – Hogyan működik a RecyclerView?**

1. **RecyclerView** létrejön az XML-ben.
2. **Az Adapter** kezeli az adatok megjelenítését az egyes elemekben.
3. **A ViewHolder** gyorsítótárazza az egyes listaelemeket.
4. **Az adatok dinamikusan töltődnek be, és az újrafelhasználás javítja a teljesítményt.**